Resum

Aquest annex es tracte de l'annex del projecte Disseny i implementació d'una llibreria Java per la simulació de sistemes de control lineals. L'annex està estructurat per diferents apartats que tracten d'elements en concret.

Per començar, es poden trobar els diagrames UML de les llibreries Java creades, per tal de poder veure les variables i mètodes de cada classe.

Tot seguit, s'explica el funcionament de l'editor de Pols i Zeros, una aplicació complexa creada mitjançant EJS, que es tracte d'un element interactiu i intuïtiu per a la definició de funcions de transferència de forma molt ràpida i que ha servit per a la creació i testeig d'aquest projecte (a més a més, de poder ser utilitzada en un futur com a base en el EJS).

A continuació s'expliquen els processos o passos que cada mètode de les classes segueix per a obtenir els resultats que s'esperen.

També s'ha afegit exemples pràctics que serveixen per demostrar el funcionament de la llibreria i poder comparar-los amb altres programaris, com ara, en el cas d'aquest projecte, MatLab.

Per tal de treure contingut a la memòria del projecte i únicament focalitzar-la en els elements o resultats importants, s'ha afegit a l'annex l'explicació de les funciona auxiliar (no principals, *protected* o privades) de la classe que serveixen també pel seu funcionament o com a extres per donar més funcionalitats al projecte.

Per acabar, s'ha creat unes aplicacions EJS que desmotren la funcionalitat i ventall de possibilitat d'aquesta llibreria. En aquest annex, es troba una guia de com funcionen aquestes aplicacions (ja que tot i ser exemples, poden ser util·litzades funcionalment) i breument s'exemplifica els seus codis d'integració per tal de que EJS pugui aprofitar-se de la llibreria del projecte.

Sumari

A. Taules de transformades

Taula de Transformades

$e(t)$	$E(s)$	$E(z)$
$u(t)$	$\frac{1}{s}$	$\frac{z}{z-1}$
t	$\frac{1}{s^2}$	$\frac{Tz}{(z-1)^2}$
$\frac{t^2}{2}$	$\frac{1}{s^3}$	$\frac{T^2z(z+1)}{2(z-1)^3}$
t^{k-1}	$\frac{(k-1)!}{s^k}$	$\lim_{a\to 0}(-1)^{k-1}\frac{\partial^{k-1}}{\partial a^{k-1}}\left[\frac{z}{z-e^{-aT}}\right]$
e^{-at}	$\frac{1}{s+a}$	$\frac{z}{z-e^{-aT}}$
te^{-at}	$\frac{1}{(s+a)^2}$	$\frac{Tze^{-aT}}{(z-e^{-aT})^2}$
t^ke^{-at}	$\frac{(k-1)!}{(s+a)^k}$	$(-1)^k\frac{\partial^k}{\partial a^k}\left[\frac{z}{z-e^{-aT}}\right]$
$1-e^{-at}$	$\frac{a}{s(s+a)}$	$\frac{z\left(1-e^{-aT}\right)}{(z-1)(z-e^{-aT})}$
$t-\frac{1-e^{-at}}{a}$	$\frac{a}{s^2(s+a)}$	$\frac{z\left[\left(aT-1+e^{-aT}\right)z+\left(1-e^{-aT}-aTe^{-aT}\right)\right]}{a(z-1)^2(z-e^{-aT})}$
$1-(1+at)e^{-at}$	$\frac{a^2}{s(s+a)^2}$	$\frac{z}{z-1}-\frac{z}{z-e^{-aT}}-\frac{aTe^{-aT}z}{(z-e^{-aT})^2}$
$e^{-at}-e^{-bt}$	$\frac{b-a}{(s+a)(s+b)}$	$\frac{\left(e^{-aT}-e^{-bT}\right)z}{\left(z-e^{-aT}\right)\left(z-e^{-bT}\right)}$
$\sin(at)$	$\frac{a}{s^2+a^2}$	$\frac{z\sin(aT)}{z^2-2z\cos(aT)+1}$
$\cos(at)$	$\frac{s}{s^2+a^2}$	$\frac{z(z-\cos(aT))}{z^2-2z\cos(aT)+1}$
$\frac{1}{b}e^{-at}\sin(bt)$	$\frac{1}{(s+a)^2+b^2}$	$\frac{1}{b}\left[\frac{ze^{-aT}\sin(bT)}{z^2-2ze^{-aT}\cos(bT)+e^{-2aT}}\right]$
$e^{-at}\cos(bt)$	$\frac{s+a}{(s+a)^2+b^2}$	$\frac{z^2-e^{aT}\cos(bT)z}{z^2-2ze^{-aT}\cos(bT)+e^{-2aT}}$
$1-e^{at}\left(\cos(bt)+\frac{a}{b}\sin(bt)\right)$	$\frac{a^2+b^2}{s\left[(s+a)^2+b^2\right]}$	$\frac{z(Az+B)}{(z-1)(z^2-2ze^{-aT}\cos(bT)+e^{-2aT})}$
		$A=1-e^{-aT}\left(\cos(bT)+\frac{a}{b}\sin(bT)\right)$
		$B=e^{-2aT}+e^{-aT}\left(\frac{a}{b}\sin(bT)-\cos(bT)\right)$
$\frac{1}{ab}+\frac{e^{-at}}{a(a-b)}+\frac{e^{-bt}}{b(b-a)}$	$\frac{1}{s(s+a)(s+b)}$	$\frac{(Az+B)z}{\left(z-e^{-aT}\right)\left(z-e^{-bT}\right)(z-1)}$

Taula A.1. Taula de transformades.

B. Diagrames UML

B.1. Diagrames UML de funcions de transferència i representació interna dins les classes

B.1.1. Diagrama UML general de la classe principal i classes derivades

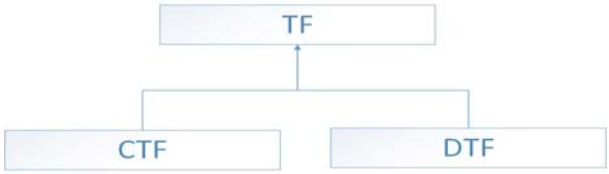


Fig. B.1. Diagrama UML general de la classe principal i classes derivades.

B.1.2. Diagrama UML dels elements de la classe TF

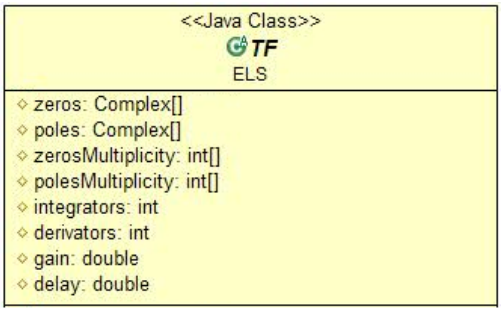


Fig. B.2. Diagrama UML dels elements de la classe TF.

B.1.3. Diagrama UML dels elements estàtics i finals de la classe TF

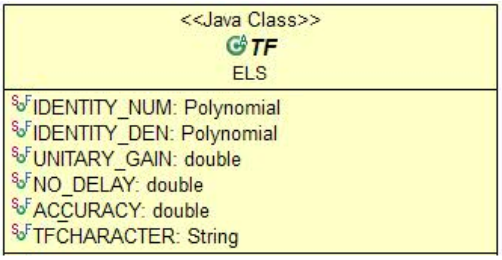


Fig. B.3. Diagrama UML dels elements estàtics i finals de la classe TF.

B.1.4. Diagrama UML d'altres elements de la classe TF.

Fig. 5.4. Diagrama UML d'altres elements de la classe TF.

B.1.5. Diagrama UML d'elements propis de les classes CTF i DTF

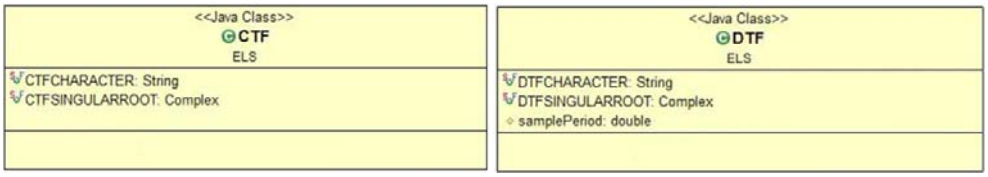


Fig. 5.5. Diagrama UML d'elements propis de les classes CTF i DTF.

B.2. Diagrames UML de contructors

B.2.1. Diagrama UML dels constructors de la classe TF

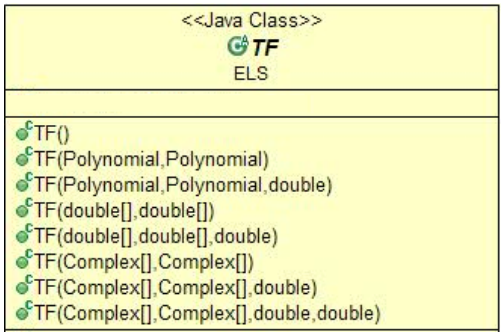


Fig. 6.1. Diagrama UML dels constructors de la classe TF.

B.2.2. Diagrama UML dels constructors de la classe CTF

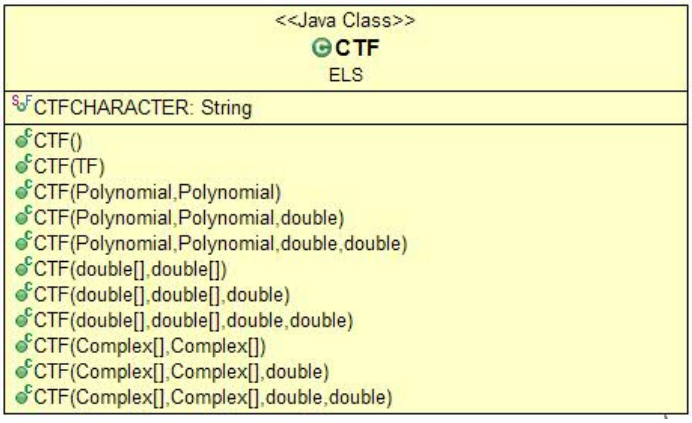


Fig B.4. Diagrama UML dels constructors de la classe CTF.

B.2.3. Diagrama UML dels constructors de la classe DTF

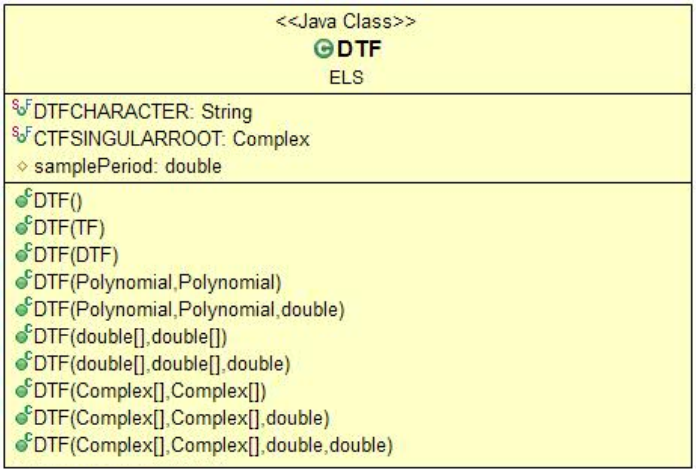


Fig B.5. Diagrama UML dels constructors de la classe DTF.

B.3. Diagrames UML mètodes per definir, modificar i eliminar elements

B.3.1. Diagrama UML dels mètodes de definició i obtenció d'element de l'estructura

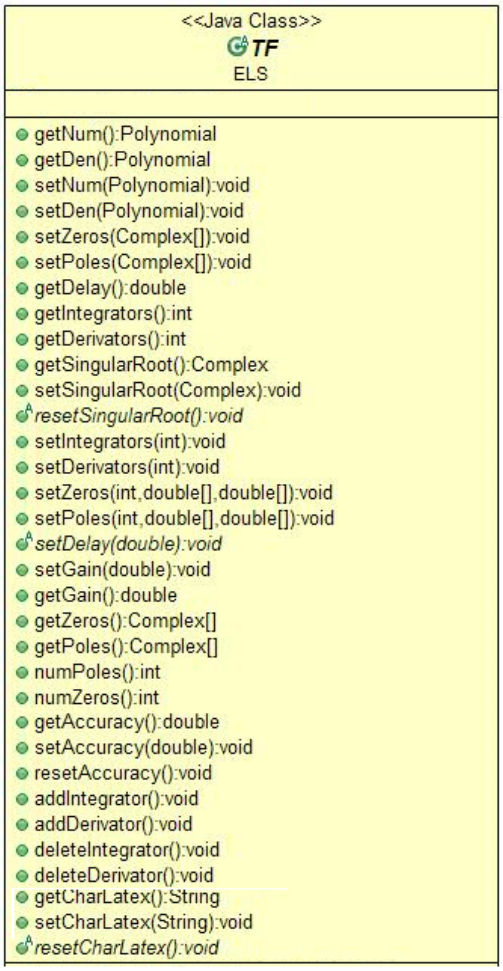


Fig. B.6. Diagrama UML dels mètodes de definició i obtenció d'element de l'estructura.

B.3.2. Diagrama UML dels mètodes de modificació i eliminació d'elements de l'estructura

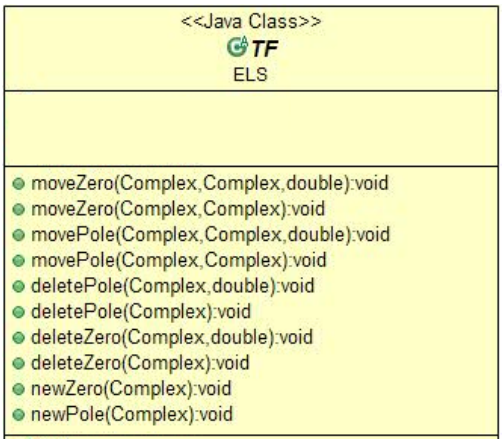


Fig. B.7. Diagrama UML dels mètodes de modificació i eliminació d'elements de l'estructura.

B.3.3. Diagrama UML dels mètodes de definició propis de la classe CTF

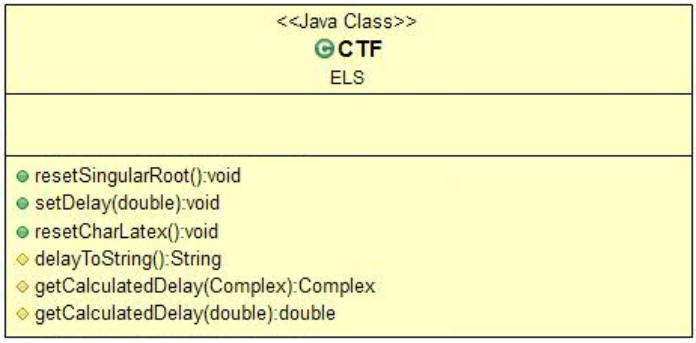


Fig. B.8. Diagrama UML dels mètodes de definició propis de la classe CTF.

B.3.4. Diagrama UML dels mètodes de definició propis de la classe DTF

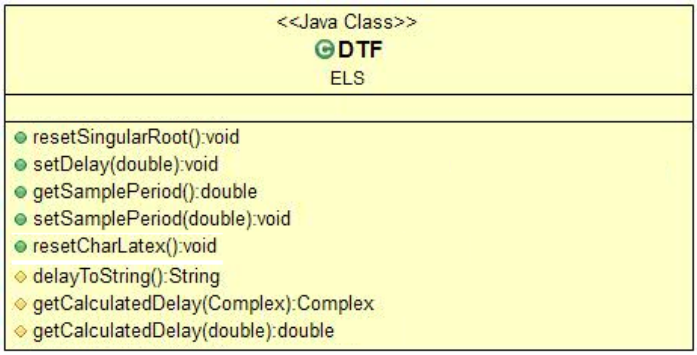


Fig. B.9. Diagrama UML dels mètodes de definició propis de la classe DTF.

B.4. Diagrames UML de Funcions de transferència definides

B.4.1. Diagrama UML dels mètodes d'obtenció de funcions de transferència definides

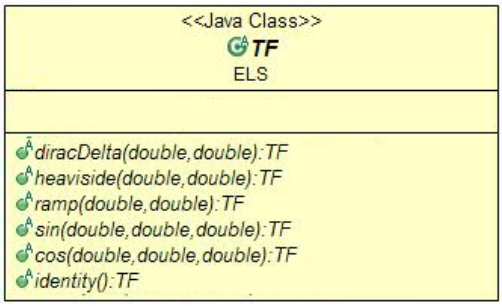


Fig. B.10. Diagrama UML dels mètodes d'obtenció de funcions de transferència definides.

B.4.2. Diagrama UML dels mètodes d'obtenció de funcions de transferència a CTF

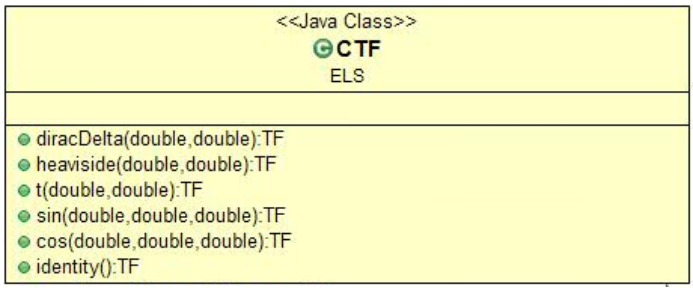


Fig. B.11. Diagrama UML dels mètodes d'obtenció de funcions de transferència a CTF.

B.4.3. Diagrama UML dels mètodes d'obtenció de funcions de transferència a DTF

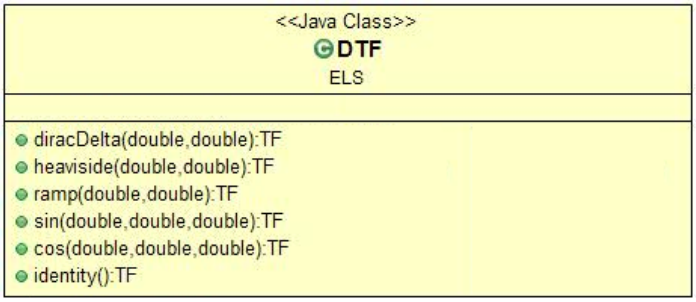


Fig. B.12. Diagrama UML dels mètodes d'obtenció de funcions de transferència a DTF.

B.5. Diagrames UML de sistemes formats per funcions de transferència

B.5.1. Diagrama UML dels mètodes càlcul de sistemes

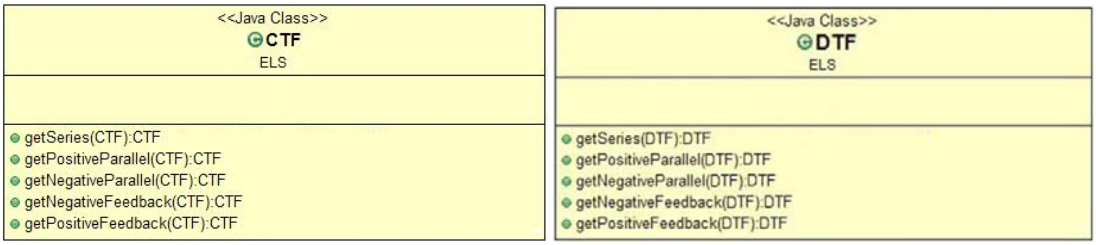


Fig. B.13. Diagrama UML dels mètodes càlcul de sistemes.

B.6. Diagrrames UML de verificació de l'estabilitat

B.6.1. Diagrama UML dels mètodes de verificació de l'estabilitat

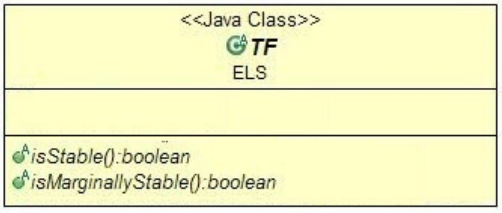


Fig. B.14. Diagrama UML dels mètodes de verificació de l'estabilitat.

B.6.2. Diagrama UML dels mètodes de verificació de l'estabilitat a la classe CTF

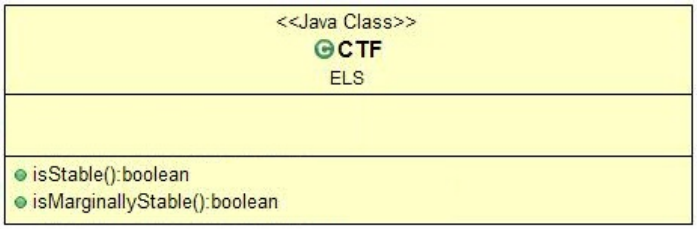


Fig. B.15. Diagrama UML dels mètodes de verificació de l'estabilitat a la classe CTF.

B.6.3. Diagrama UML dels mètodes de verificació de l'estabilitat a la classe DTF

Fig. B.16. Diagrama UML dels mètodes de verificació de l'estabilitat a la classe DTF.

B.7. Diagrames UML de funcions d'escriptura de funcions de transferència a EJS

B.7.1. Diagrama UML dels mètodes d'escriptura de TF a EJS

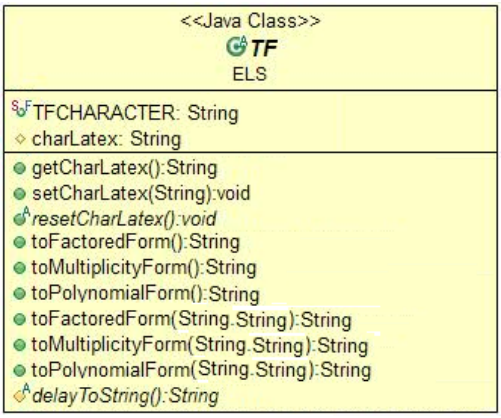


Fig. B.17. Diagrama UML dels mètodes d'escriptura de TF a EJS.

B.7.2. Diagrama UML del mètode per obtenir el retard segons el tipus d'objecte

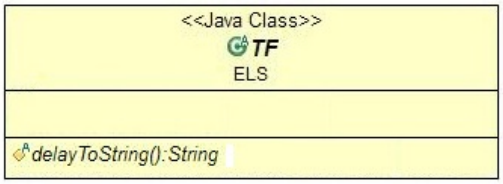


Fig. B.18. Diagrama UML del mètode per obtenir el retard segons el tipus d'objecte.

B.7.3. Diagrama UML dels mètodes d'escriptura de TF amb paràmetres i sense

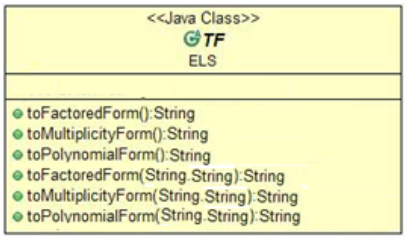


Fig. B.19. Diagrama UML dels mètodes d'escriptura de TF amb paràmetres i sense.

B.8. Diagrames UML de mètodes d'avaluació de valors

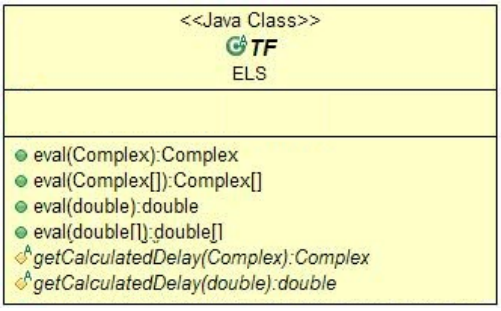


Fig. B.20. Diagrama UML dels mètodes d'avaluació de valors.

B.9. Diagrames UML de funcions de coeficients de posició, velocitat i acceleració.

B.9.1. Diagrama UML de mètodes d'obtenció de coeficients de posició velocitat i acceleració

Fig. B.21. Diagrama UML de mètodes d'obtenció de coeficients de posició velocitat i acceleració.

B.10. Diagrames UML de resposta freqüencial

B.10.1. Diagrama UML dels mètodes d'obtenció de dades de la resposta freqüencial

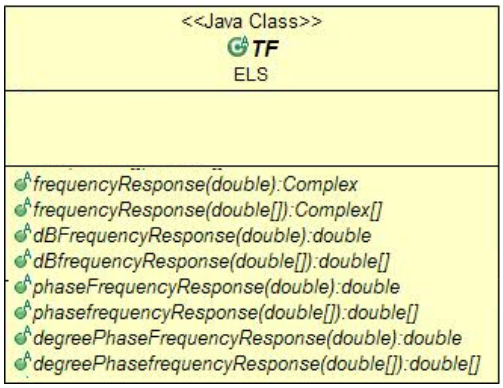


Fig. B.22. Diagrama UML dels mètodes d'obtenció de dades de la resposta freqüencial.

B.10.2. Diagrama UML de mètodes de la resposta freqüencial a CTF i DTF

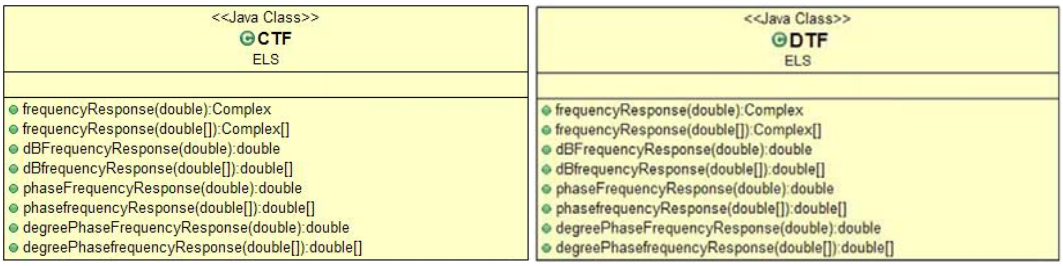


Fig. B.23. Diagrama UML de mètodes de la resposta freqüencial a CTF i DTF.

B.11. Diagrames UML de transformacions

B.11.1. Diagrama UML dels mètodes de descomposició de fraccions simples

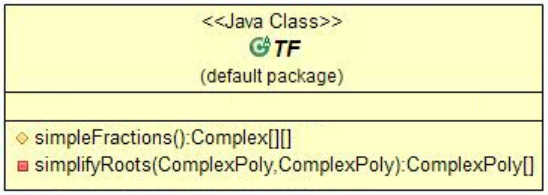


Fig. B.24. Diagrama UML dels mètodes de descomposició de fraccions simples.

B.11.2. Diagrama UML del mètode de transformació bilineal

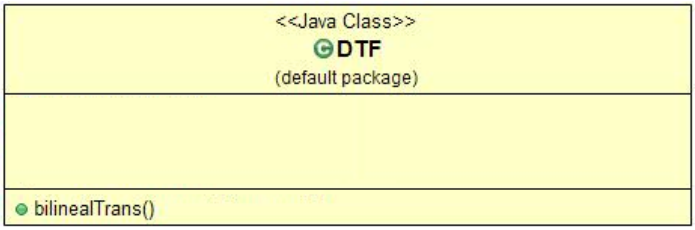


Fig. B.25. Diagrama UML del mètode de transformació bilineal.

B.11.3. Diagrama UML del mètode de la antitransformació bilineal

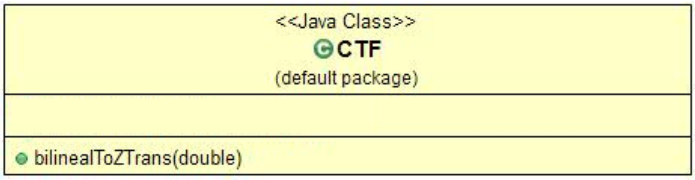


Fig. B.26. Diagrama UML del mètode de la antitransformació bilineal.

B.12. Diagrames UML de funcions del diagrama de Bode

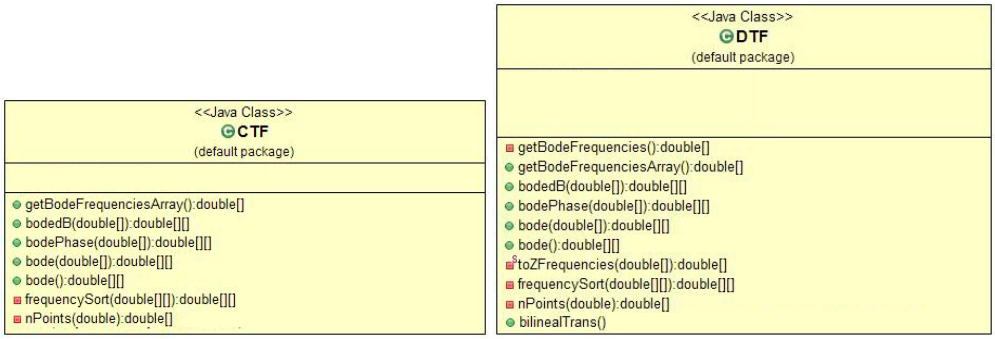


Fig. B.27. Diagrama UML dels mètodes del diagrama de Bode.

B.13. Diagrames UML de funcions del diagrama de Bode asimptòtic

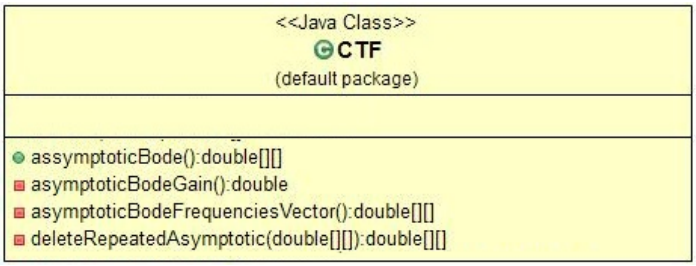


Fig. B.28. Diagrama UML dels mètodes de representació del Bode asimptòtic.

B.14. Diagrames UML de funcions del diagrama de Nyquist

Fig. B.29. Diagrama UML dels mètode de representació del diagrama de Nyquist.

B.15. Diagrames UML de funcions de Nichols

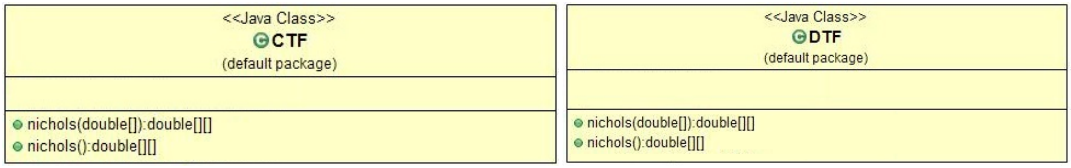


Fig. B.30. Diagrama UML dels mètode de representació del diagrama de Nichols.

B.16. Diagrames UML de funcions de resposta temporal

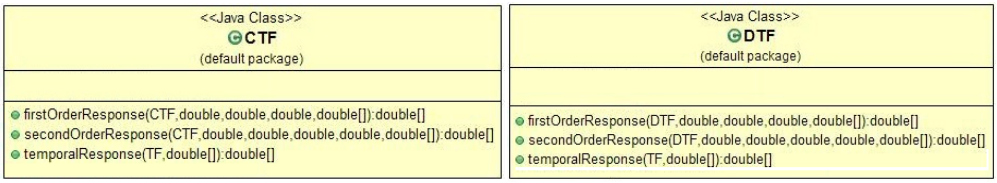


Fig. B.31. Diagrama UML dels mètode de representació de la resposta temporal.

C. Funcionament de l'Editor de pols i zeros

C.1. Afegir pols i zeros

Inicialment, els sistemes que l'editor controla estan formats per elements unitaris (sense pols ni zeros). Per introduir un pol o zero al sistema, s'ha de fer *click* sobre la creu o cercle que hi ha situats a la part superior esquerre de l'editor i arrossegar aquest element al centre de l'editor. Si es selecciona un element blau, s'introduirà en el sistema anteriorment anomenat CTF 1 i si es selecciona un vermell s'introduirà en el sistema CTF 2.

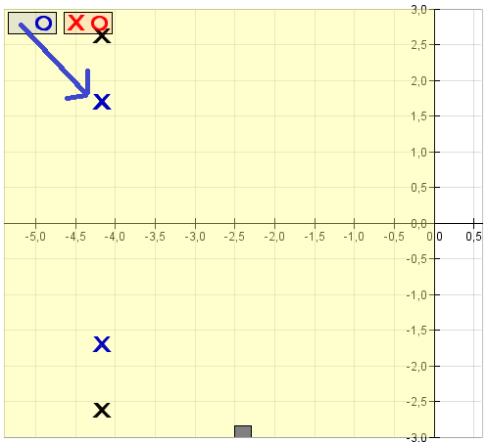


Fig. C.1. Acció d'afegir pols i zeros dins l'editor.

Com es pot observar, l'acció demostrada a la figura anterior introdueix dos pols al sistema, degut a que a l'introduir un pol en el sistema, al passar per la part dels complexes, és necessari que existeixi el seu element complexa conjugat. Per tant, l'editor ja preveu això i afegeix també aquest element de forma automàtica.

Quan es deixa el botó del ratolí en una posició, aquest element introduït queda situat i definit amb els valors sobre els quals s'ha deixat de sobre l'eix. Es dir, si es deixa el pol sobre el punt (-1, 1), l'editor afegirà automàticament dos pols a (-1 +1j) i a (-1 -1j).

En cas de que s'afegeixi un pol o zero sobre l'eix real, al no necessitar el seu complexa conjugat, únicament és crearà un element d'aquell tipus. Es a dir, si es situa sobre l'eix, només s'estarà afegint una arrel, si es situa a l'espai dels complexes, se n'afegirà dos.

C.2. Eliminar pols i zeros

Per eliminar pols i zeros del sistema, el que s'ha de fer es arrossegar aquest element a eliminar cap al requadre d'on ha sortit (requadre superior esquerre). L'editor l'eliminarà i també eliminarà, en cas d'existir, el seu element complexa conjugat vinculat a ell.

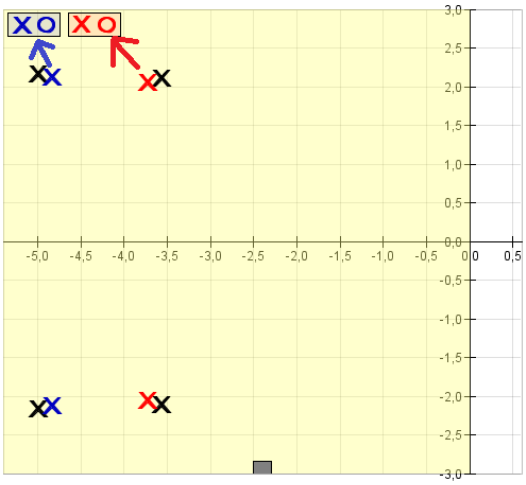


Fig. C.2. Acció d'eliminar pols i zeros dins l'editor.

Existeix també la possibilitat de fer doble "click" sobre l'arrel que es vol eliminar i apareixerà una finestra amb un botó amb l'etiqueta "Eliminar Raiz" que permet eliminar l'arrel seleccionada (eliminant també l'arrel complexa conjugada en cas d'existir).

C.3. Modificar pols i zeros

Existeixen dos maneres de modificar els valors dels pols i dels zeros.

La primera forma, es tracta de arrossegar el pol o zero que interessa modificar fins a la nova posició que interessa, fent "click" al damunt, aguantant mentre s'arrossega i deixant anar sobre la nova posició.

En el posicionament arrossegant existeix un indicador, que apareix a la part inferior esquerre, que permet saber en quina posició es col·loca més exactament l'arrel que s'està editant.

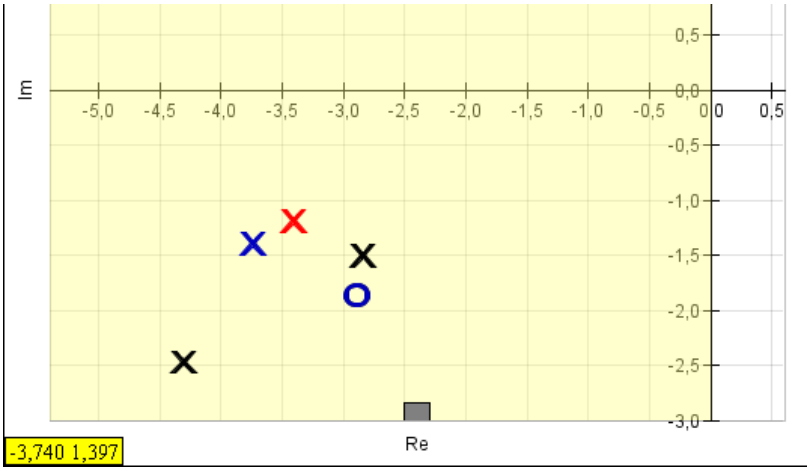


Fig. C.3. Indicador de posició dels pols i zeros durant la seva manipulació.

El segon mètode, és tracta d'un mètode més precís en el posicionament i posiciona el pol o zero en ell lloc on es desitja. Per aquest mètode, cal fer doble "click" sobre l'arrel a modificar i apareixerà una finestra on es podrà definir la nova posició del pol i zero a on es vol col·locar i tot seguit, seleccionar el boto "Ok".

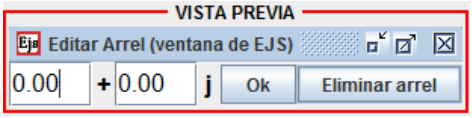


Fig. C.4 Finestra de posicionament dels pols i zeros.

D'aquesta forma, l'arrel es posicionarà exactament en el punt on es desitja i, en cas d'existir una arrel complexa conjugada vinculada a aquesta arrel, l'editor també la modificarà a la mateixa posició amb part imaginària conjugada.

Una cosa que cal remarcar, és que en la creació d'arrels, si es situava una arrel sobre l'eix real, únicament s'afegia una arrel al sistema. En aquest cas, a l'estar les dos arrels en el sistema, si es col·loquen sobre l'eix real, les dues arrels continuaran existint sobre aquest eix.

C.4. Posicionament en punts característics

Per tal de fer més fàcil el posicionament de pols i zeros dins l'editor, s'ha creat també uns elements que en punts característics es puguin col·locar arrels de forma més senzilla.

Els primers punts característics es tracten de la col·locació d'arrels sobre els eixos. Per tal de facilitar la col·locació d'arrels sobre els eixos, s'ha introduït un marge que permet, en cas de no col·locar l'arrel exactament sobre l'eix, situar-lo automàticament sobre aquest.

Per fer-ho, només cal acostar l'arrel sobre qualsevol dels dos eixos i apareixerà un relleu que indica que l'arrel s'està deixant dins aquest marge de valors, en el qual l'editor situarà l'arrel (o les arrels, en cas de complexa conjugat) sobre l'eix:

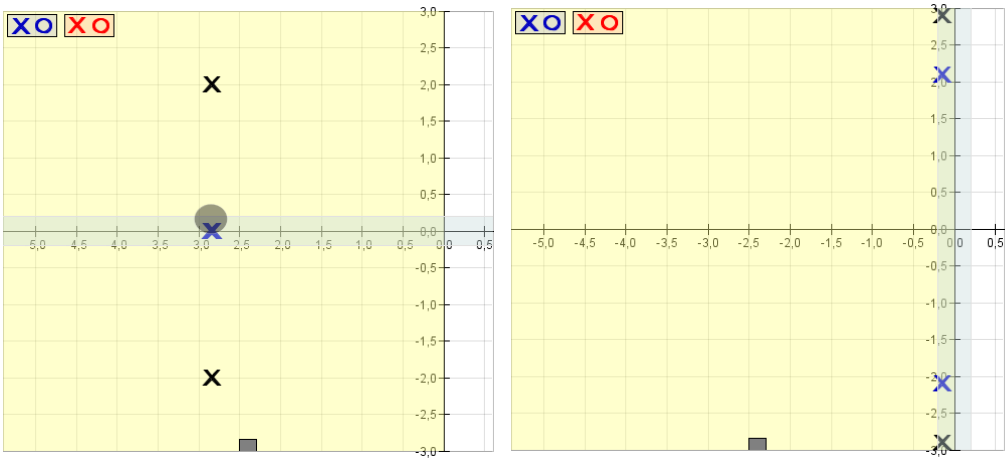


Fig. C.5. Marges de col·locació d'elements sobre els eixos.

En cas de voler situar una arrel en un creuament dels eixos, apareix també un cercle d'influència que indica que l'arrel serà col·locada en el centre del punt:

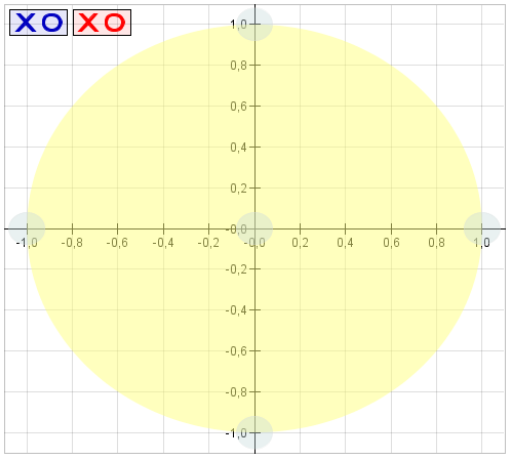


Fig. C.6. Punts característics dels editors.

C.5. Accions limitades

Pel que fa a les arrels que es troben sobre l'eix real, si no disposen de cap arrel vinculada a elles que pugui ser la seva complexa conjugada sobre l'espai dels complexes, no se'ls permet col·locar-les fora d'aquest eix. En cas de provar-ho, l'arrel retornarà sobre l'eix real a la posició on s'hagi deixat:

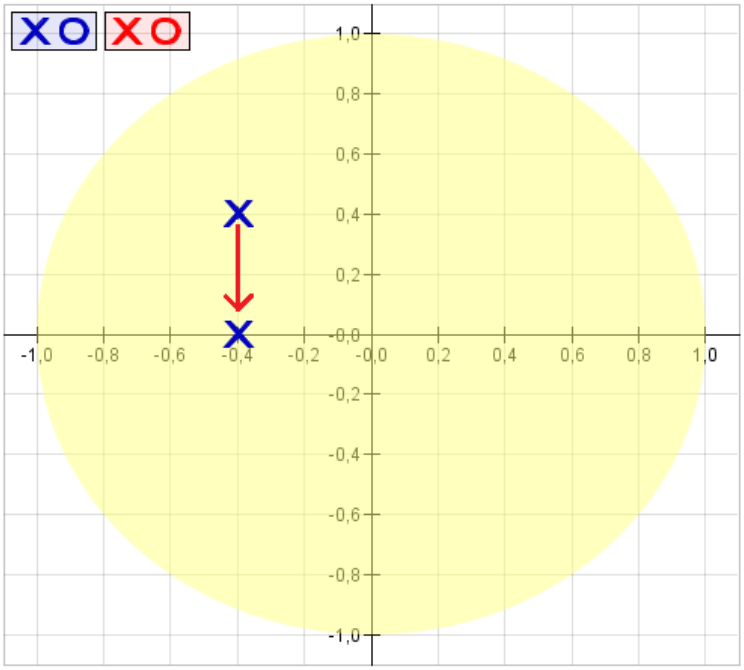


Fig. C.7. Posicionament d'una arrel única sobre l'espai dels complexes.

Si es situa sobre la caixa superior esquerre, s'estarà portant a terme l'acció d'eliminar aquesta arrel.

Quan es modifiquen aquestes arrels que hi ha sobre l'eix dels reals, al seu voltant apareix una zona d'influència que, en cas de ser ocupada per una altre arrel única sobre l'eix real, les dos arrels que han entrat en influencia, passen a vincular-se i a poder ser complexes conjugades:

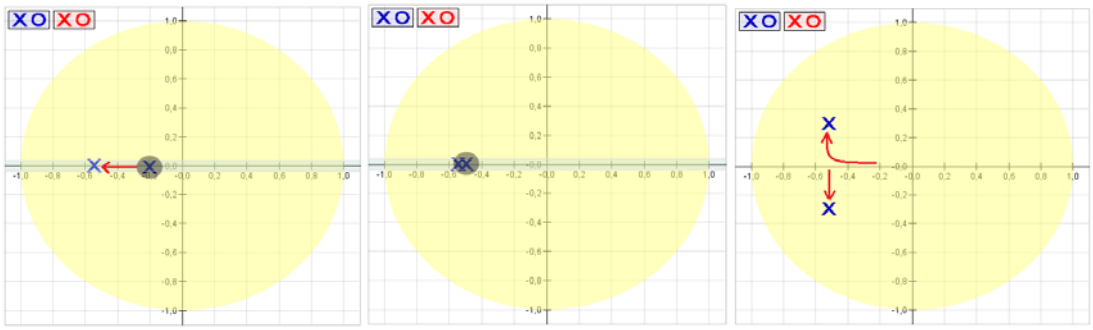


Fig. C.8. Pas d'arrels reals sobre l'espai dels complexes.

Per tornar-les a separar, només cal portar-les sobre l'eix real altre vegada i moure-les per sobre aquest fins a separar-los de la zona d'influencia, es a dir, realitzar els passos anteriors de forma inversa.

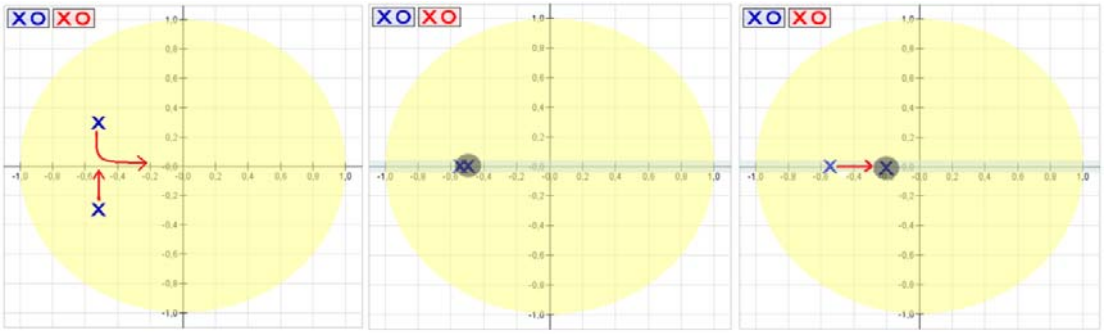


Fig. C.9. Pas d'arrels de l'espai complexa sobre l'eix dels reals.

D. Explicació dels mètodes de modificació d'arrels

D.1. Procés seguit pels mètodes *move*

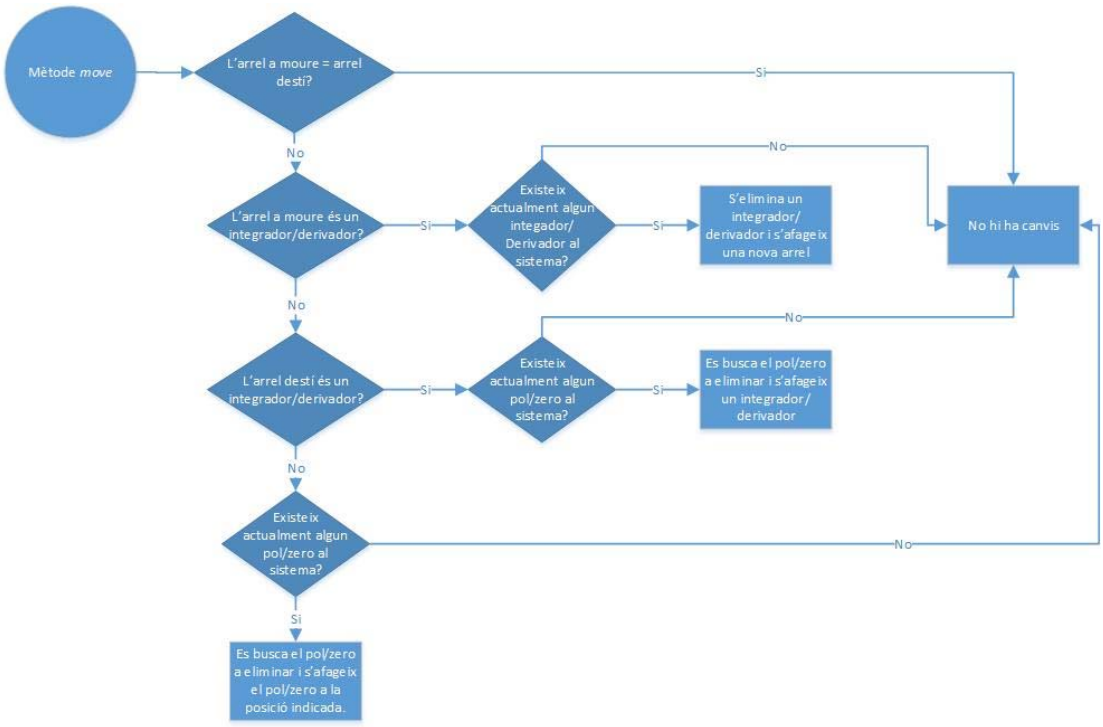


Fig. D.1. Procés seguit pels mètodes "move".

D.2. Procés seguit pels mètodes *delete*

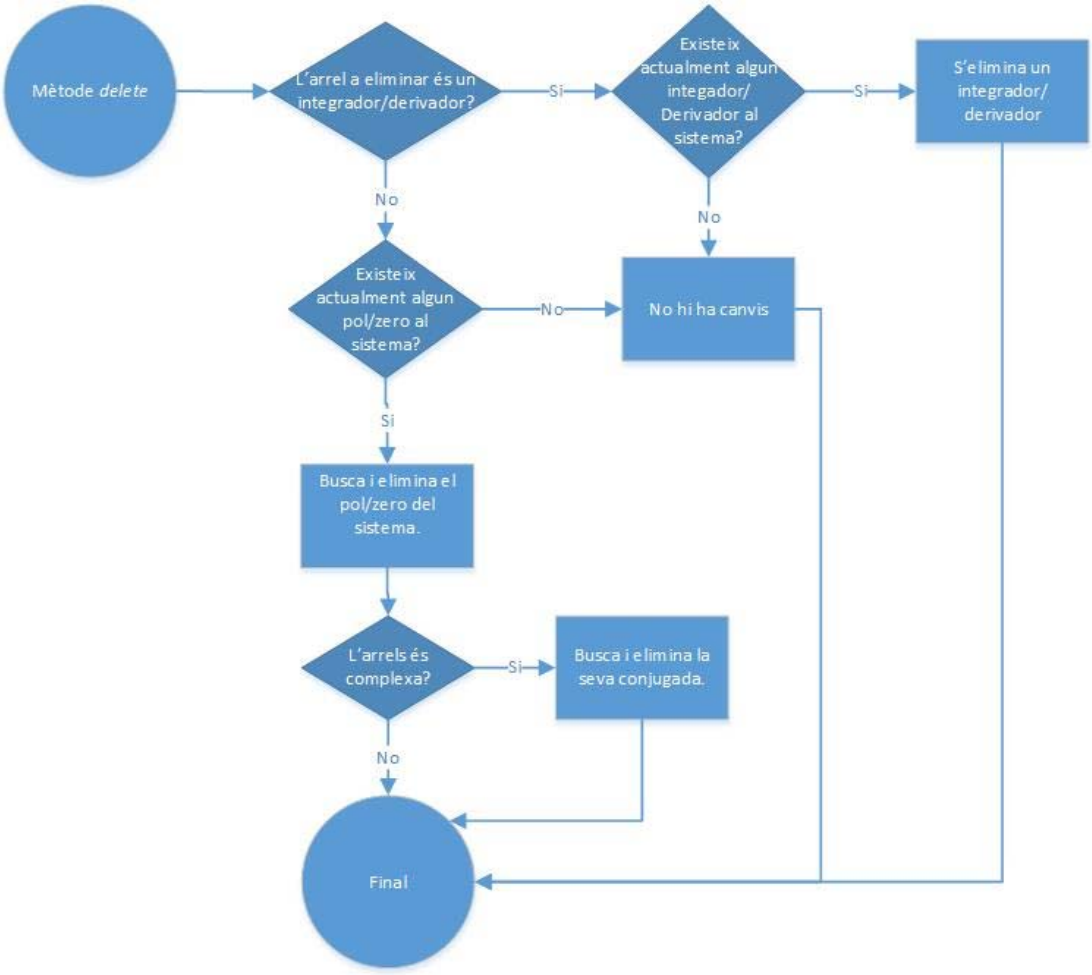


Fig. D.2. Procés seguit pels mètodes "delete".

D.3. Procés seguit pels mètodes *new*

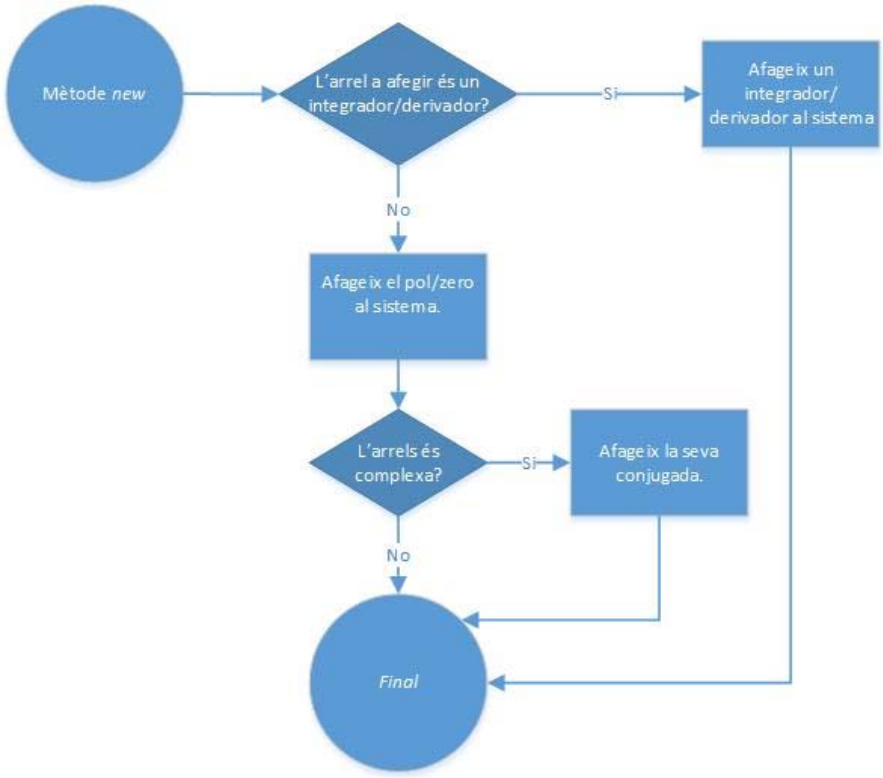


Fig. D.3. Procés seguit pels mètodes "new".

D.4. Procés seguits pel càlcul d'un sistema en sèrie

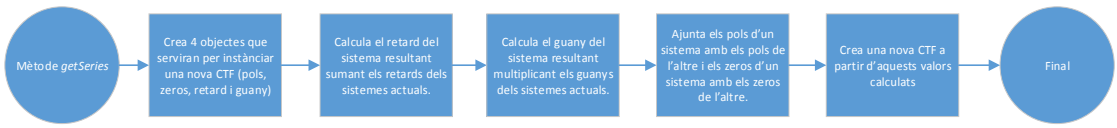


Fig. D.4. Passos seguits pel càlcul d'un sistema en sèrie.

D.5. Procés seguits pel càlcul d'un sistema en paral·lel positiu

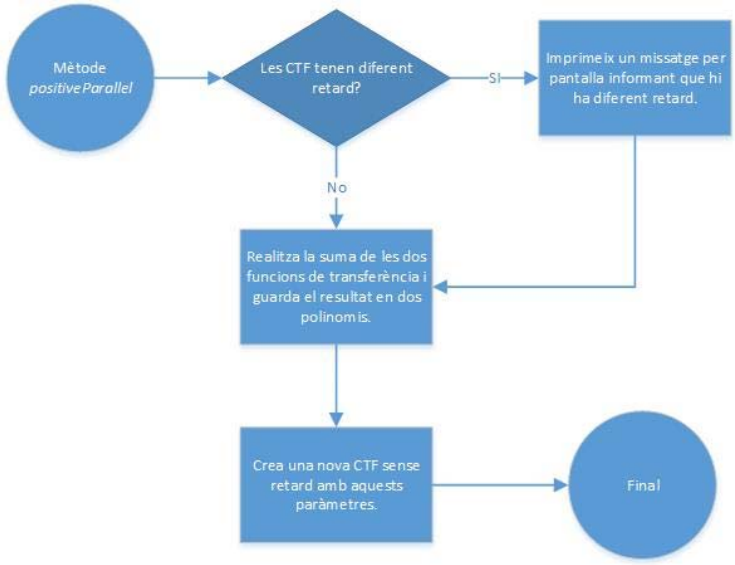


Fig. D.5. Passos seguits pel càlcul d'un sistema en paral·lel positiu.

D.6. Procés seguit pel mètode d'escriptura factoritzada

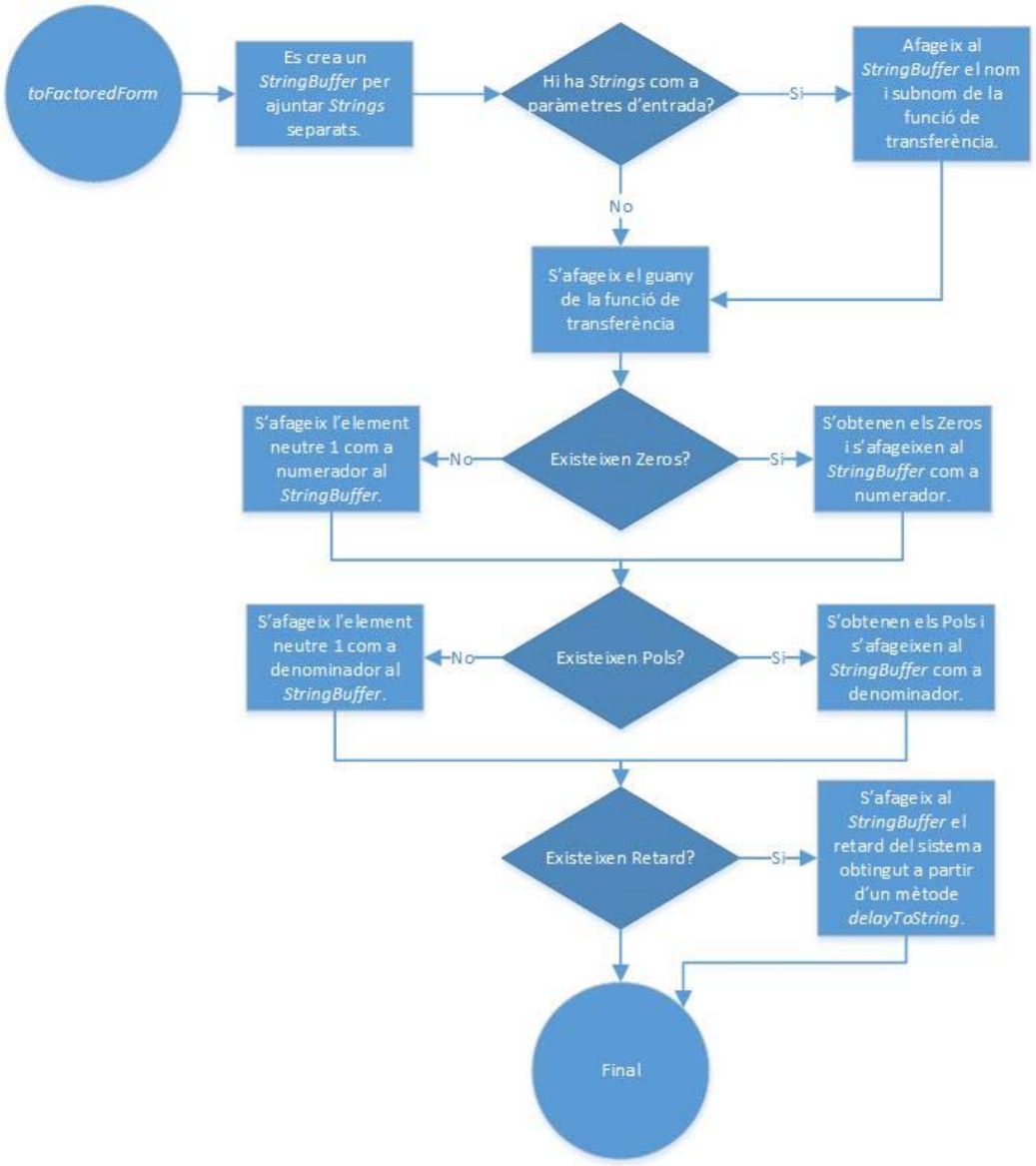


Fig. D.6. Procés seguit pel mètode d'escriptura factoritzada.

D.7. Procés seguit pel mètode d'escriptura en la representació polinòmica

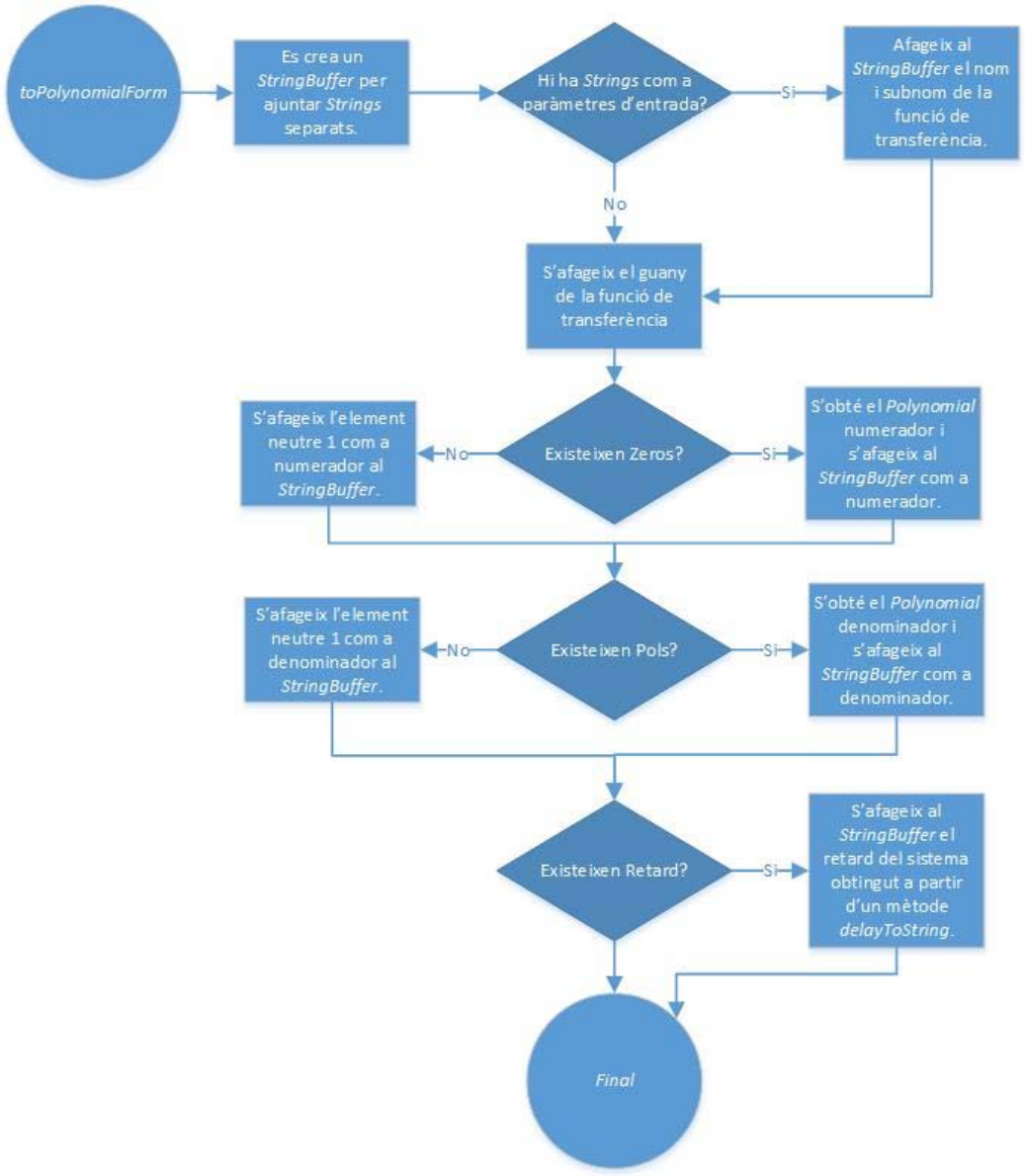


Fig. D.7. Procés seguit pel mètode d'escriptura en la representació polinòmica.

D.8. Procés seguit pel mètode d'escriptura de la representació amb multiplicitat

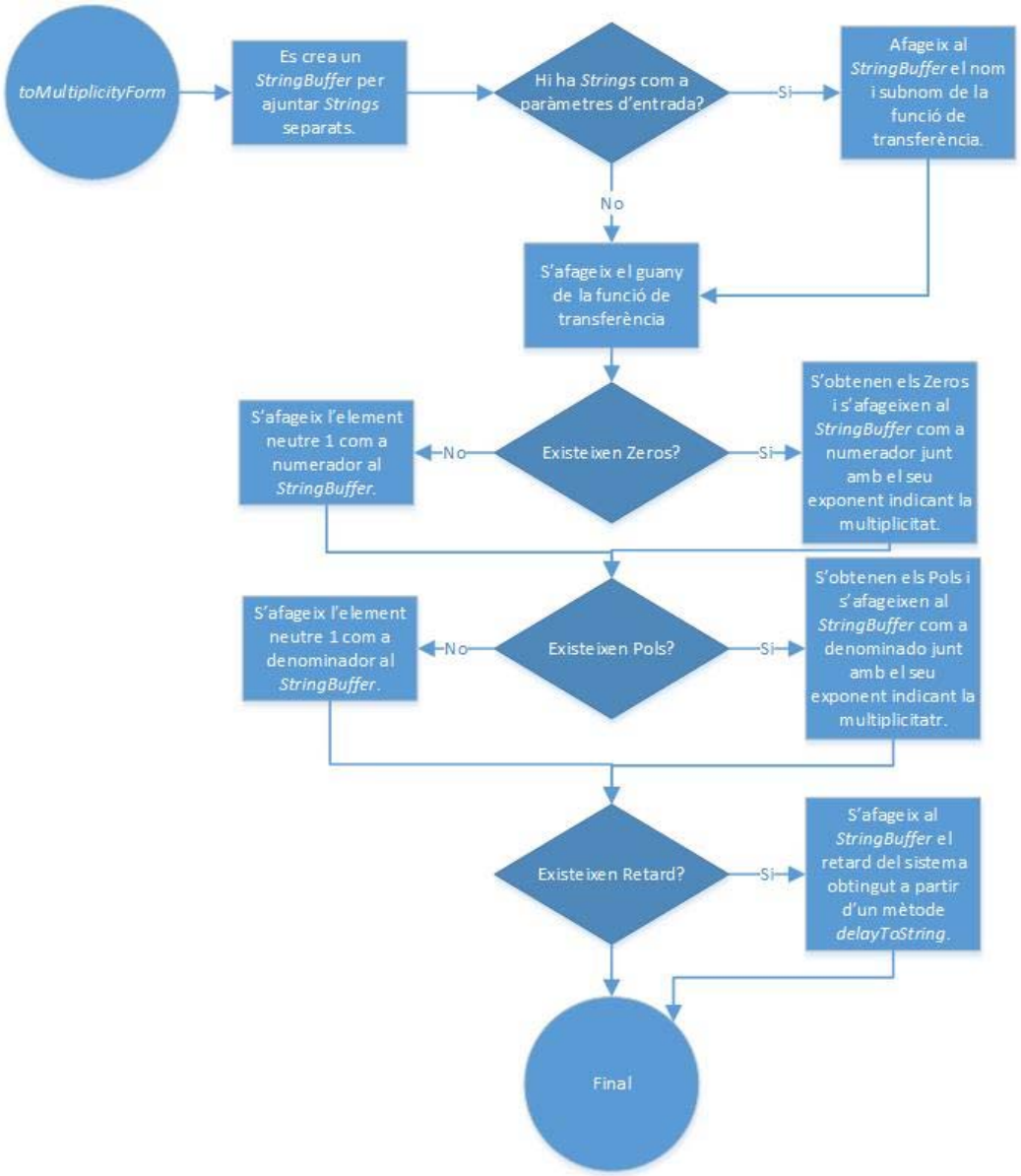


Fig. D.8. Procés seguit pel mètode d'escriptura de la representació amb multiplicitat.

D.9. Procés seguit pels mètodes d'avaluació de valors

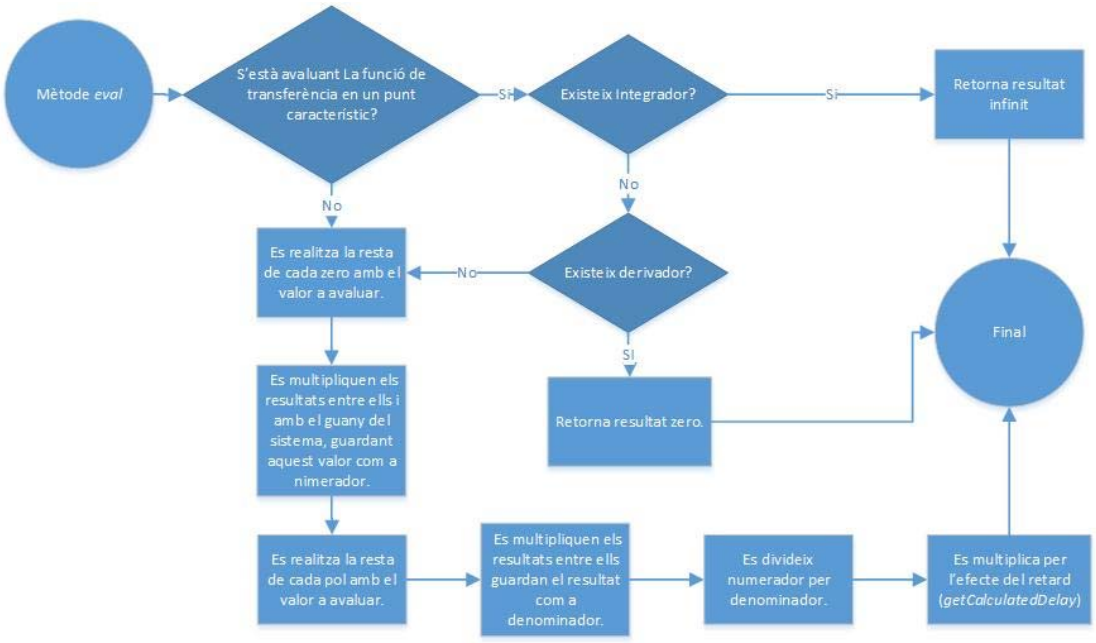


Fig. D.9. Procés seguit pels mètodes d'avaluació de valors.

D.10. Procés seguit per l'obtenció del coeficient de posició

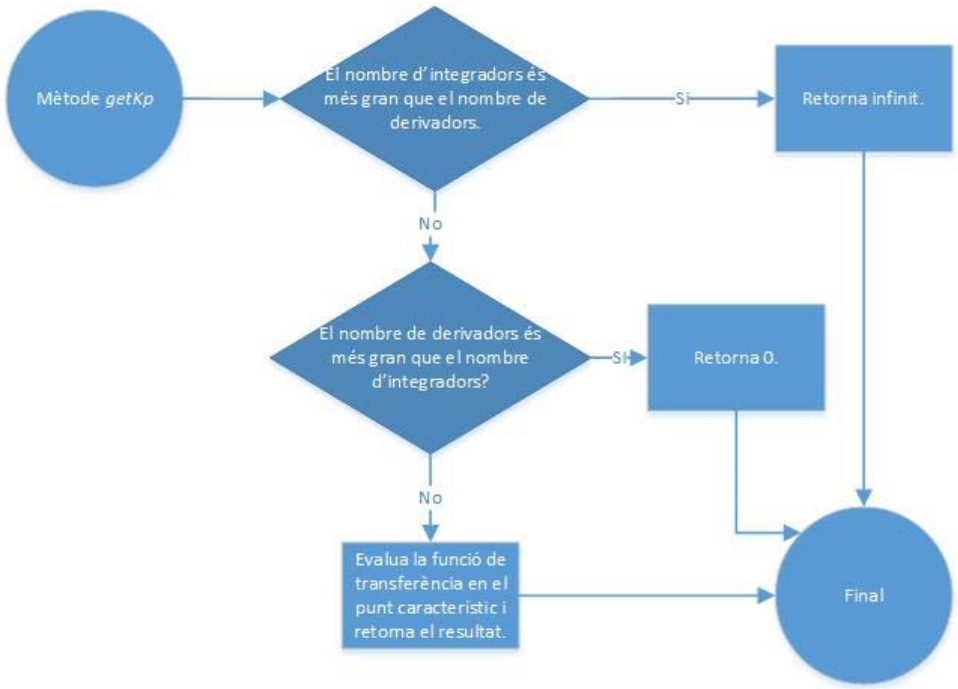


Fig. D.10. Procés seguit per l'obtenció del coeficient de posició.

D.11. Procés seguit per l'obtenció del coeficient de velocitat

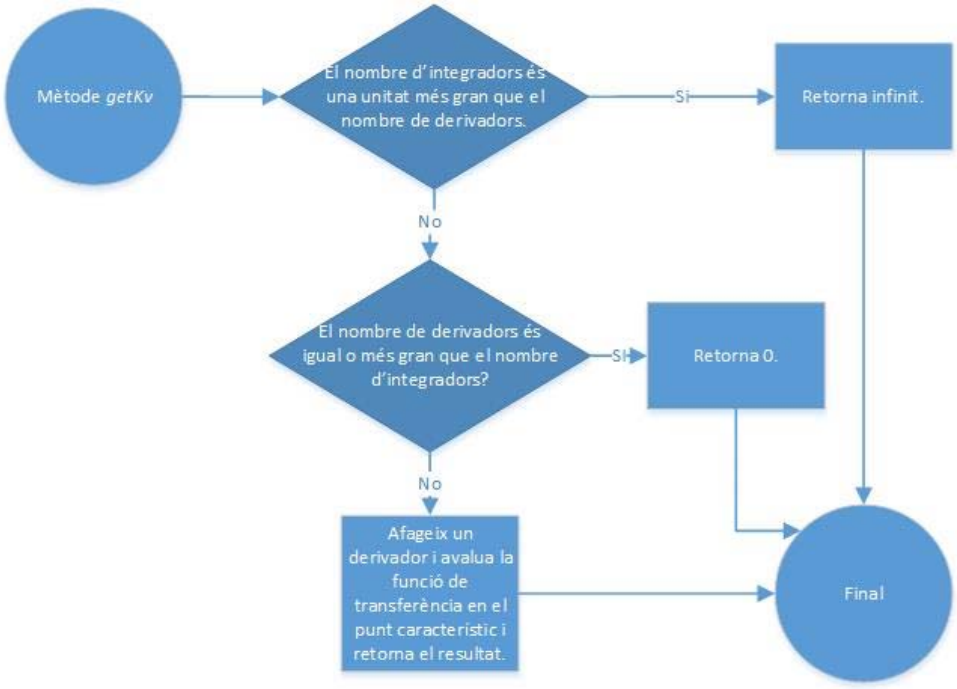


Fig. D.11. Procés seguit per l'obtenció del coeficient de velocitat.

D.12. Procés seguit per obtenir el coeficient d'acceleració

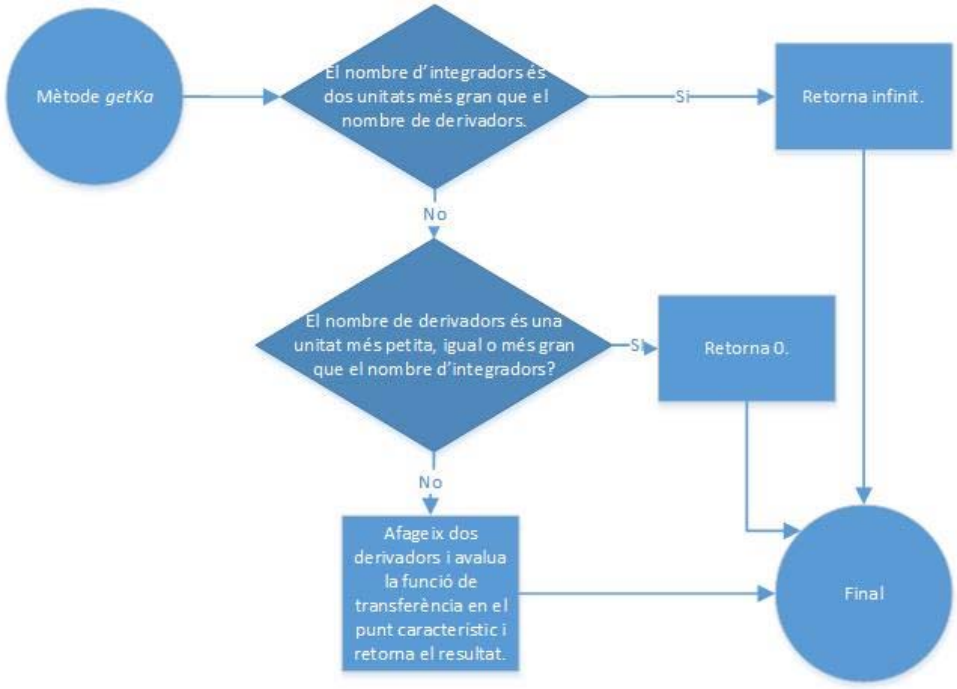


Fig. D.12. Procés seguit per obtenir el coeficient d'acceleració.

D.13. Procés seguit per obtenció de la resposta freqüencial

Fig. D.13. Procés seguit per obtenció de la resposta freqüencial.

D.14. Procés seguit l'obtenció de la resposta freqüencial en dB

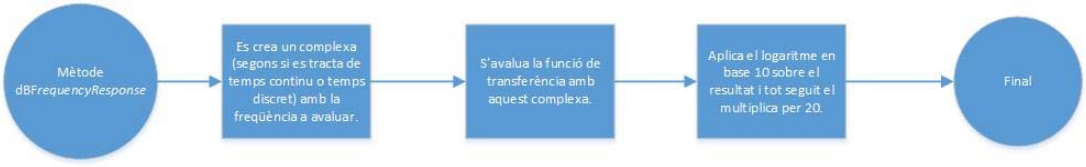


Fig. D.14. Procés seguit l'obtenció de la resposta freqüencial en dB.

D.15. Procés seguit per l'obtenció de la fase de la resposta freqüencial

Fig. D.15. Procés seguit per l'obtenció de la fase de la resposta freqüencial.

E. Exemples teòrics i pràctics

E.1. Exemple teòric del mètode *move*

Un exemple seria el cas següent, en que es disposa de l'equació de transferència indicada i es desitja canviar de valor del pol situat a -3,02 del sistema a la posició -4 indicada a les següents figures:

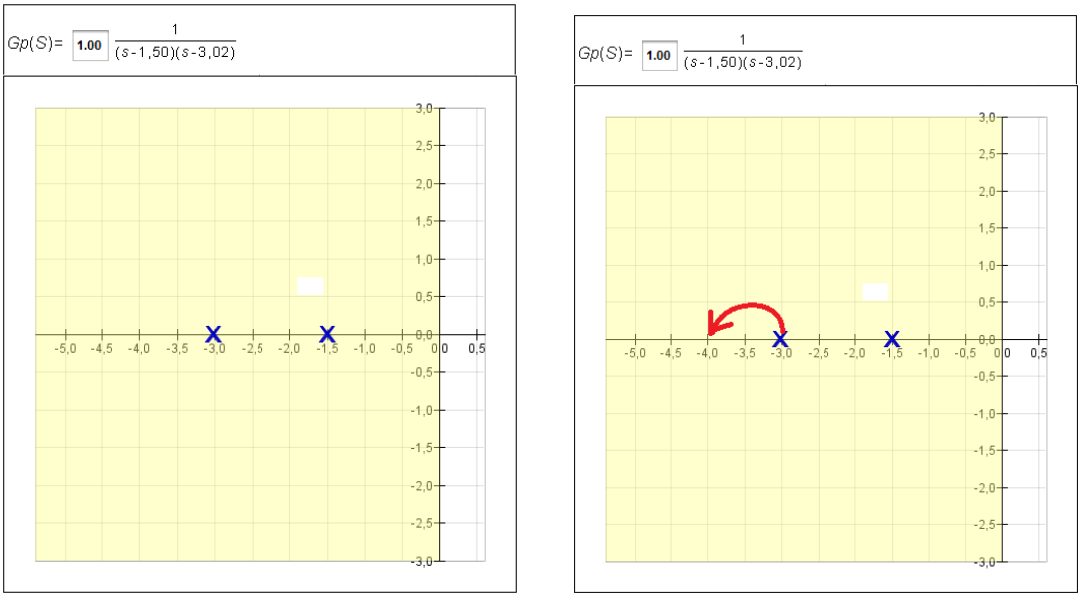


Fig. E.1. Moviment d'un pol.

En aquest cas, aquest mètodes obtindria les arrels del denominador de la funció de transferència fent servir un mètode de la classe *Polynomial*. Aquestes arrels obtingudes, com s'ha comentat abans, no tindrien exactament el valor de -1,5 i 3,02. Sinó que s'obtindran unes arrels (per exemplificar hipotèticament) de valor aproximat de -1,5000000000000000 ± 0,0000001 i -3,0200000000000000 ± 0,0000001. Per tant, si com a paràmetre d'entrada es dona un valor de -3,02 exacte i el mètode retorna una arrel de valor -3,020000213647523 doncs en el moment de comparar aquest valor amb -3,02, no es considerarien iguals.

En aquest moment, és quan entra en joc el tercer paràmetre d'entrada. Aquest valor indica dins a quin marge absolut s'ha de trobar l'arrel. Suposant el cas en que aquest paràmetre tingués un valor de 0,5, si s'indiqués que el pol està situat a -3 i s'estableix aquest marge de

0,5, el pol a trobar haurà d'estar dins els valors de [-2,5;-3,5] i per tant, a estar a dins aquest rang, seleccionarà l'arrel que es desitja.

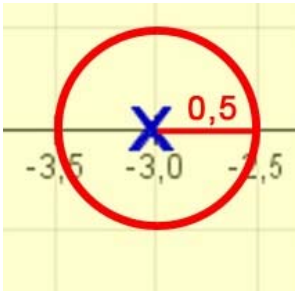


Fig. E.2. Exemple de marge de tolerància.

Per altre banda, podria existir el cas de que dos o més arrels es trobessin molt pròximes al valor indicat, amb la qual cosa, un paràmetre de marge molt gran englobaria varis pols i si es precisa exactitud en la selecció de l'arrel, podria induir a un error i que s'acabi movent una arrel equivocada.

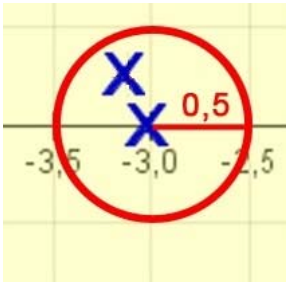


Fig. E.3. Exemple de pols dins el cercle d'influència.

Per aquesta raó, s'ha d'ajustar aquets paràmetre el màxim possible perquè seleccioni sempre l'arrel correcte, però també s'ha de mantenir un marge suficient per a que cobreixi l'error de càlcul de JAVA.

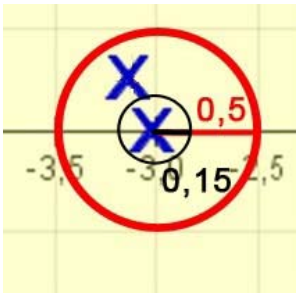


Fig.E.4. Exemple de reducció del marge d'influència.

Per aquest motiu s'ha creat al inici de la classe un element estàtic final que té de nom *ACCURACY* i que estableix un valor de marge que compleix força correctament les condicions descrites al paràgraf anterior. S'ha de tenir en compte que per temes docents no és necessària una precisió molt alta, amb la qual cosa se sol treballar a nivell del segon o tercer decimal. Degut que l'error de càlcul de Java en totes les proves realitzades s'ha mogut aproximadament en el setè decimal, un valor de *ACCURACY* de 0,0002 és considera suficient.

Si s'observa el diagrama UML de la classe (a l'annex indicat), es pot observar que els mètodes "*move"* també existeixen únicament amb dos paràmetres d'entrada del tipus *Complex* (es a dir, sense el tercer paràmetre que estableix el marge d'error absolut). Aquest mètodes estableixen per defecte un valor de marge indicat per l'element estàtic final *ACCURACY* de 0,0002.

E.2. Exemple pràctic de la funció de descomposició en fraccions simples

Per a fer aquest exemple, s'ha modificat la funció simpleFractions de la classe abstracte TF de funció protected a funció públic.

El codi util·litzat ha estat el següent:

```
package ctfTest;

import els.CTF;
import els.DTF;
import els.TF;
import flanagan.complex.Complex;
import flanagan.math.Polynomial;

public class SimpleFractionsTest {
   public static void main(String args[]){
            System.err.println("");
                        System.err.println("----------------------");
```

```
                System.err.println("TF() simpleFractions:");
        System.err.println("(for this example the function simpleFractions have been set as a public)");
                System.err.println("");

        double[] coeff = {-1,1};
        Polynomial numer = new Polynomial(coeff);
        coeff = new double[4];
        coeff[0] = 0;
        coeff[1] = 1;
        coeff[2] = 2;
        coeff[3] = 1;
                Polynomial denom = new Polynomial(coeff);
            DTF tf2 = new DTF(numer,denom);
        System.err.println(tf2.toPolynomialForm());
        tf2.setSamplePeriod(1);
        //tf2.setAccuracy(1);
        Complex[][] sp = tf2.simpleFractions();
        System.err.println(" ");
        System.err.println("Matrix result:");
        for(int i=0;i<sp.length;i++){
           for(int t=0;t<sp[1].length;t++){
              System.err.print(sp[i][t].getReal());
              System.err.print(", ");
            }
           System.err.println("");
        }

   }

}
```

El resultat obtingut, fent servir la mateixa funció de tranfarència que la de l'apartat del projecte 16.1 és el següent:

$$X(s) = \frac{s-1}{s^3 + 2s^2 + s}$$

```
----------------------
TF() simpleFractions:
(for this example the function simpleFractions have been set as a public)

 \frac{z-1,00}{z^{3}+\,2,00z^{2}+\,1,00z+\,0,00}

Matrix result:
0.0, -1.0, -1.0,
1.0, 1.0, 2.0,
1.0, 2.0, 2.0,
-1.0, 1.0, 2.0,
BUILD SUCCESSFUL (total time: 0 seconds)
```

Aconseguint la matriu indicada que teòricament s'obtindria com a resultat:

$$Matiu[0][num\ pols] = \{0, \quad -1, \quad -1\}$$

$$Matiu[1][num\ pols] = \{1, \quad 1, \quad 2\}$$

$$Matiu[2][num\ pols] = \{1, \quad 2, \quad 2\}$$

$$Matiu[3][num\ pols] = \{-1, \quad 1, \quad 2\}$$

E.3. Exemple pràctic de la funció transformada bilineal

El codi util·litzat per aquest exemple pràctic ha estat el següent:

```
package ctfTest;
import els.CTF;
import els.DTF;
import els.TF;
import flanagan.complex.Complex;
import flanagan.math.Polynomial;
public class BilinealTransTest {
   public static void main(String args[]){
            System.err.println("");
                        System.err.println("----------------------");
                        System.err.println("CTF() bilinealToZTrans:");
                        System.err.println("");

            double[] coeff = {2,0,3};
            Polynomial numer = new Polynomial(coeff);
            coeff = new double[3];
            coeff[0] = 3;
            coeff[1] = 2;
            coeff[2] = 1;
                        Polynomial denom = new Polynomial(coeff);

                  CTF tf2 = new CTF(numer,denom);
           System.err.println(tf2.toPolynomialForm());
           //tf2.setSamplePeriod(1);
           //tf2.setAccuracy(1);
           DTF bTF = tf2.bilinealToZTrans(2);
           System.err.println(" ");
           System.err.println("DTF result:");
           System.err.println(bTF.toPolynomialForm());
   }
```

La funció de transferència util·litzada ha estat la mateixa que la de l'apartat 16.2 del projecte.

El resultat obtingut, és el següent (coincidint amb el de l'exemple teòric):

```
CTF() bilinealToZTrans:

 \frac{3,00s^{2}+\,0,00s+\,2,00}{s^{2}+\,2,00s+\,3,00}
ELS.TF - causalityTest: Numerator degree must be smaller than Denominator. This transfer function is not casual.
ELS.TF - causalityTest: Numerator degree must be smaller than Denominator. This transfer function is not casual.

DTF result:
 \frac{2,50z^{2}+\,1,00z+\,2,50}{z^{2}-2,00z^{1}+\,3,00}
BUILD SUCCESSFUL (total time: 0 seconds)
```

E.4. Exemple pràctic de la funció antitransformada bilineal

Fent servir el codi següent:

```
package ctfTest;
import els.CTF;
import els.DTF;
import els.TF;
import flanagan.complex.Complex;
import flanagan.math.Polynomial;
public class BilinealTransToCTFTest {
   public static void main(String args[]){
```

```
            System.err.println("");
                        System.err.println("----------------------");
                        System.err.println("DTF() bilineaTrans:");
                        System.err.println("");
            double[] coeff = {2.5,1,2.5};
            Polynomial numer = new Polynomial(coeff);
            coeff = new double[3];
            coeff[0] = 3;
            coeff[1] = -2;
            coeff[2] = 1;
                        Polynomial denom = new Polynomial(coeff);
                DTF tf2 = new DTF(numer,denom);
            tf2.setSamplePeriod(2);
            System.err.println(tf2.toPolynomialForm());
            //tf2.setSamplePeriod(1);
            //tf2.setAccuracy(1);
            CTF bTF = tf2.bilinealTrans();
            System.err.println(" ");
            System.err.println("CTF result:");
            System.err.println(bTF.toPolynomialForm());
    }

}
```

Es pot comprovar com el resultat obtingut és l'esperat i indicat a l'apartat 16.3 del projecte:

$$TF(z) = \frac{3z^2 + 2}{z^2 + 2z + 3}$$

```
----------------------
DTF() bilineaTrans:

 \frac{2,50z^{2}+\,1,00z+\,2,50}{z^{2}-2,00z^{1}+\,3,00}
2.0
ELS.TF - causalityTest: Numerator degree must be smaller than Denominator. This transfer function is not casual.
ELS.TF - causalityTest: Numerator degree must be smaller than Denominator. This transfer function is not casual.

CTF result:
 \frac{3,00s^{2}+\,0,00s+\,2,00}{s^{2}+\,2,00s+\,3,00}
BUILD SUCCESSFUL (total time: 0 seconds)
```

E.5. Exemple pràctic del vector de freqüències a representar en un diagrama de Bode

Per a demostrar l'exemple pràctic es farà servir la funció de transferència següent:

$$TF(s) = \frac{1}{(s + 4 + 2j) \cdot (s + 4 - 2j)}$$

Amb la cual cosa, s'hauria d'oobtenir les següents freqüències tal i com s'ha indicat a l'apartat 17.1.1 del projecte:

$$TF(s) = \frac{1}{(s + 4 + 2j) \cdot (s + 4 - 2j)} \qquad \rightarrow \qquad f_1 = \sqrt{4^2 + 2^2} \qquad f_2 = \sqrt{4^2 + (-2)^2}$$

$$f_1 = f_2 = \sqrt{20}$$

$$f[101] = \left(\frac{\sqrt{20}}{100}, \quad \frac{\sqrt{20}}{100} \cdot 100^{\frac{1}{50}}, \quad \frac{\sqrt{20}}{100} \cdot 100^{\frac{2}{50}}, \ldots, \sqrt{20}, \ldots, \sqrt{20} \cdot 100^{\frac{49}{50}}, \quad \sqrt{20} \cdot 100\right)$$

$$f[101] = (0{,}0447, 0{,}0490, \ldots, 4{,}472, \ldots, 407{,}8635, 447{,}2136)$$

Per demostrar-ho, s'ha creat el següent codi:

```
package ctfTest;
import els.CTF;
import els.DTF;
import els.TF;
import els.AuxiliarPolynomial;
import flanagan.complex.Complex;
import flanagan.math.Polynomial;
/**
 *
 * @author PC
 */
public class CTFBodeFrecuencyArrayTest {

    public static void main(String args[]){
            System.err.println("");
                    System.err.println("----------------------");
                    System.err.println("CTF() BodeFrequencyArray:");
                    System.err.println("");
            double[] coeff = {1};
            Polynomial numer = new Polynomial(coeff);
            Complex[] roots = new Complex[2];
            roots[0] = new Complex(-4,2);
            roots[1] = new Complex(-4,-2);
                    Polynomial denom = AuxiliarPolynomial.rootsToPoly(roots);

                CTF tf2 = new CTF(numer,denom);

            System.err.println(tf2.toFactoredForm());

            double[] vectorFreq = tf2.getBodeFrequenciesArray();
            System.err.println(" ");
            System.err.println("Frequencies:");
            System.err.println("Num of points = " + vectorFreq.length);
            System.err.println("[" + vectorFreq[0] + ", " + vectorFreq[1] + ",..., " + vectorFreq[(int) (vectorFreq.length/2)-1] + ", " +
vectorFreq[(int) (vectorFreq.length/2)] + ", " + vectorFreq[(int) (vectorFreq.length/2)+1] + ",..., " + vectorFreq[(int)
(vectorFreq.length-2)] + ", " + vectorFreq[(int) (vectorFreq.length-1)] + "]");

    }

}
```

El resultat obtingut, com es pot veure, ha estat el mateix que l'exemple pràctic:

```
run:

----------------------
CTF() BodeFrequencyArray:

degree of the polynomial is zero in the method ComplexPoly.roots
null returned
 \frac{1}{(s\,-4,00-2,00j\,)(s\,-4,00+\,2,00j\,)}

Frequencies:
Num of points = 101
[0.044721359549995794, 0.049035995648450195,..., 4.078636466032924, 4.472135954999596, 4.903599564845037,...,
407.8636466032938, 447.21359549996106]
BUILD SUCCESSFUL (total time: 0 seconds)
```

E.6. Exemple pràctic del vector de freqüències a representar en un diagrama de Bode assimptòtic

Per a demostrar l'exemple pràctic es farà servir la funció de transferència següent:

$$TF(s) = \frac{1}{(s+4+2j)\cdot(s+4-2j)}$$

Amb la cual cosa, s'hauria d'oobtenir les següents freqüències tal i com s'ha indicat a l'apartat 18.1.1 del projecte:

$$TF(s) = \frac{1}{(s+4+2j)\cdot(s+4-2j)} \qquad \rightarrow \qquad f_1 = \sqrt{4^2+2^2} \qquad f_2 = \sqrt{4^2+(-2)^2}$$

$$f_1 = f_2 = \sqrt{20}$$

$$f[3] = \left(\frac{\sqrt{20}}{100}, \sqrt{20}, \sqrt{20} + \frac{\sqrt{20}}{1000}, \sqrt{20}\cdot 100\right)$$

$$f[3] = (0{,}0447, \quad 4{,}4721, \quad 4{,}4766, \quad 447{,}2135)$$

Per demostrar-ho, s'ha creat el següent codi (i s'ha passat la funció d'obtenció de freqüències de mètode privat a públic):

```
package ctfTest;

import els.CTF;
import els.DTF;
import els.TF;
import els.AuxiliarPolynomial;
import flanagan.complex.Complex;
import flanagan.math.Polynomial;

/**
 *
 * @author PC
 */
public class CTFAssimptoticBodeFreqTest {

   public static void main(String args[]){


        System.err.println("");
                System.err.println("----------------------");
                System.err.println("CTF() AssimptoticBodeFrequencyArray:");
                System.err.println("");

        double[] coeff = {1};
        Polynomial numer = new Polynomial(coeff);
        Complex[] roots = new Complex[2];
        roots[0] = new Complex(-4,2);
        roots[1] = new Complex(-4,-2);
                Polynomial denom = AuxiliarPolynomial.rootsToPoly(roots);

            CTF tf2 = new CTF(numer,denom);
```

```
        System.err.println(tf2.toFactoredForm());

        double[][] vectorFreq = tf2.asymptoticBodeFrequenciesVector();
        System.err.println(" ");
        System.err.println("Frequencies:");
        System.err.println("Num of points = " + vectorFreq[0].length);
        System.err.println("[" + vectorFreq[0][0] + ", " + vectorFreq[0][1] + ", " + vectorFreq[0][2] + ", " + vectorFreq[0][3] + "]");

    }

}
```

El resultat obtingut, com es pot veure, ha estat el mateix que l'exemple pràctic:

```
run:
----------------------
CTF() AssimptoticBodeFrequencyArray:

degree of the polynomial is zero in the method ComplexPoly.roots
null returned
 \frac{1}{(s\,-4,00-2,00j\,)(s\,-4,00+\,2,00j\,)}

Frequencies:
Num of points = 4
[0.044721359549995794, 4.47213595499958, 4.47660809095458, 447.21359549995793]
BUILD SUCCESSFUL (total time: 0 seconds)
```

E.7. Exemple pràctic de comparació amb un Bode de MatLab a la classe CTF.

Tot seguit es realitzarà un exemple de resultat obtingut mitjançant una aplicació creada en EJS en comparació amb el resultat obtingut de MatLab per la mateixa funció de transferència.

Fent servir la funció de transferència següent:

$$TF(s) = \frac{2s+2}{s^2+5s+3} = 2 \cdot \frac{s+1{,}000}{(s+0.697)(s+4{,}302)}$$

S'obté el següent a l'aplicació creada en EJS:

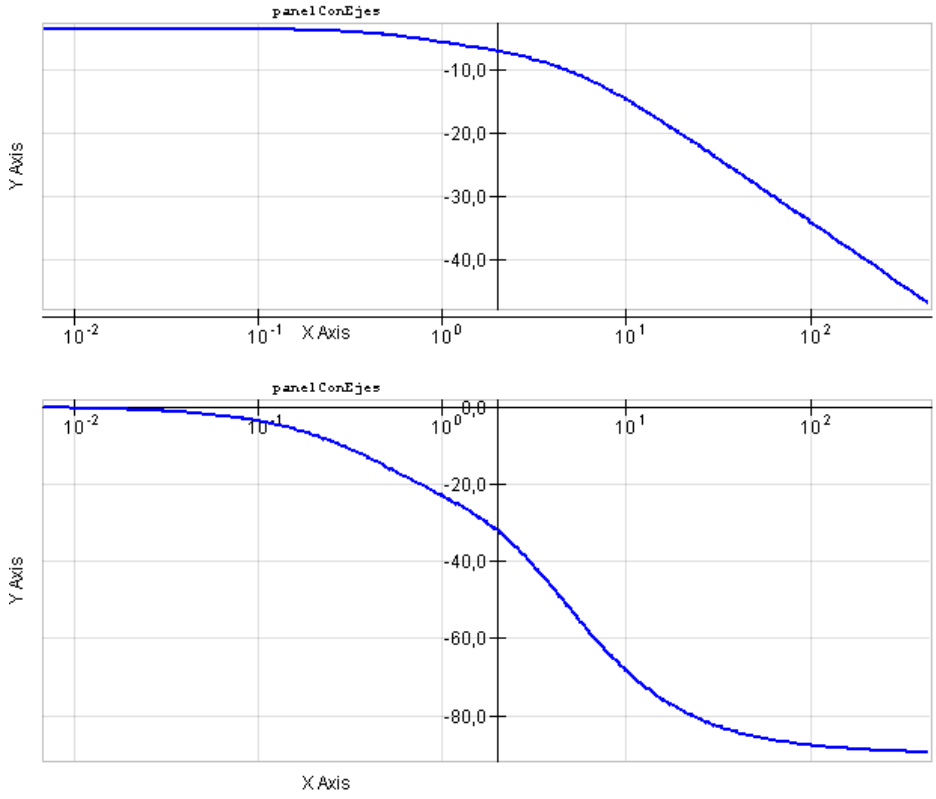


Fig. 17.4. Diagrama de Bode resultant amb EJS 1.

Pel que fa a MatLab, s'obté el següent resultat:

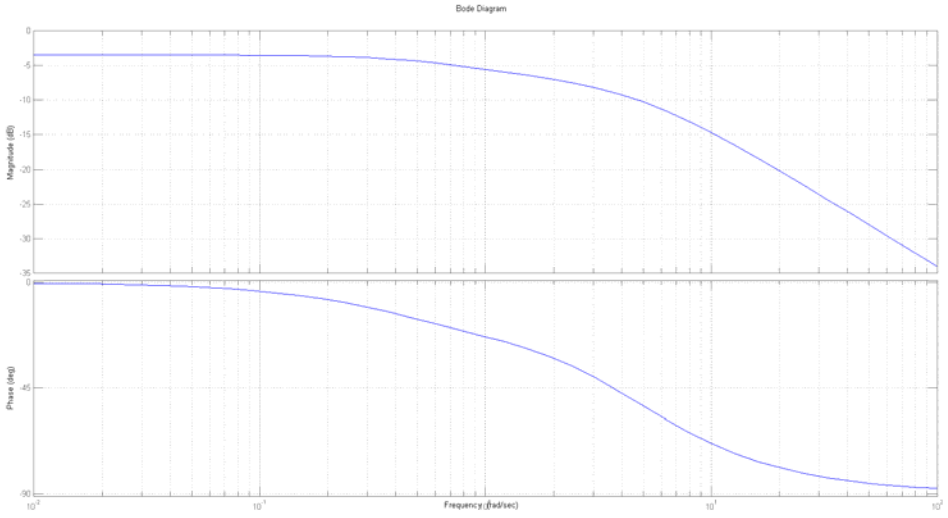


Fig. 17.5. Diagrama de Bode resultat amb MatLab 1.

Fent servir una altre funció de transferència:

$$TF(s) = \frac{s - 0{,}5}{s + 4}$$

S'obté el següent a l'aplicació creada en EJS:

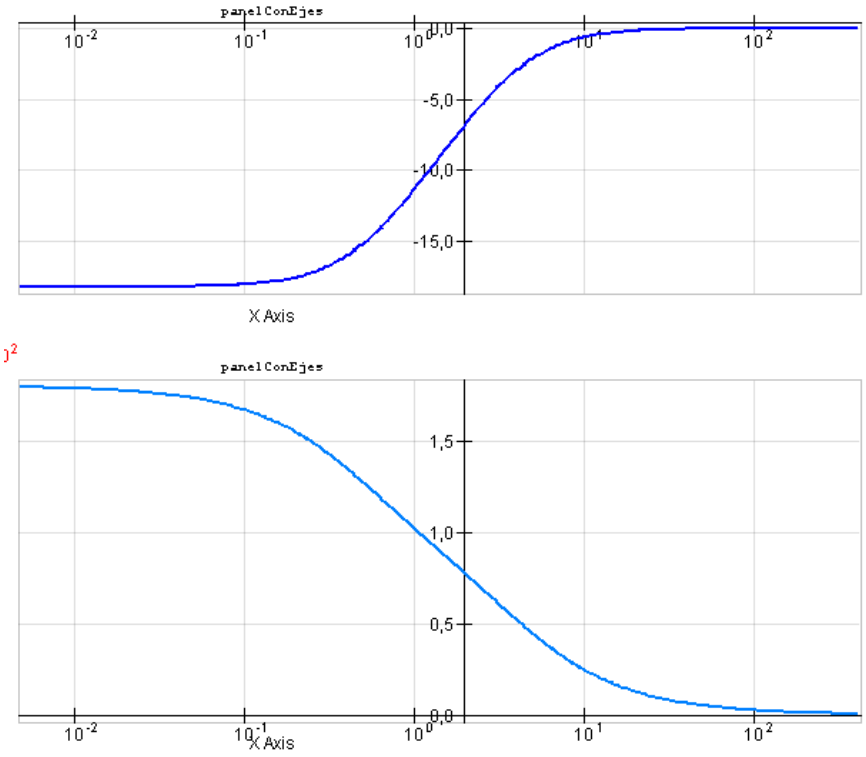


Fig. 17.4. Diagrama de Bode resultant amb EJS 2.

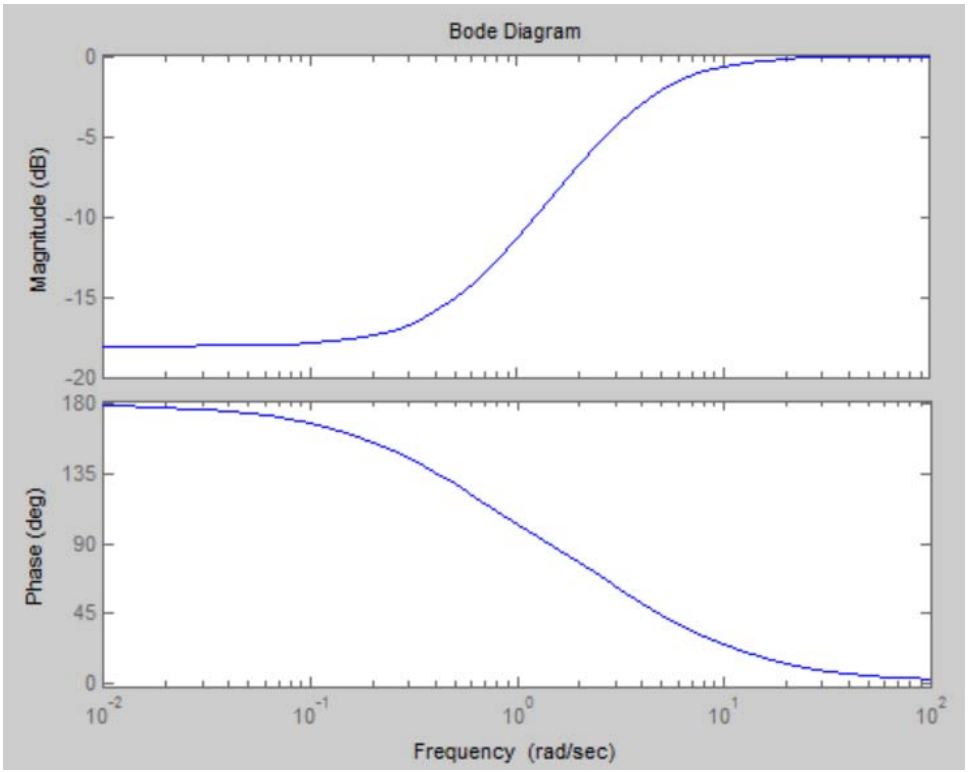


Fig. 17.5. Diagrama de Bode resultat amb MatLab 2.

Fent servir una altre funció de transferència:

$$TF(s) = \frac{s \cdot (s + 0{,}5 - 1{,}5j)(s + 0{,}5 + 1{,}5j)}{(s + 5)(s + 3 - 2j)(s + 3 + 2j)(s + 1 - j)(s + 1 + j)}$$

S'obté el següent a l'aplicació creada en EJS:

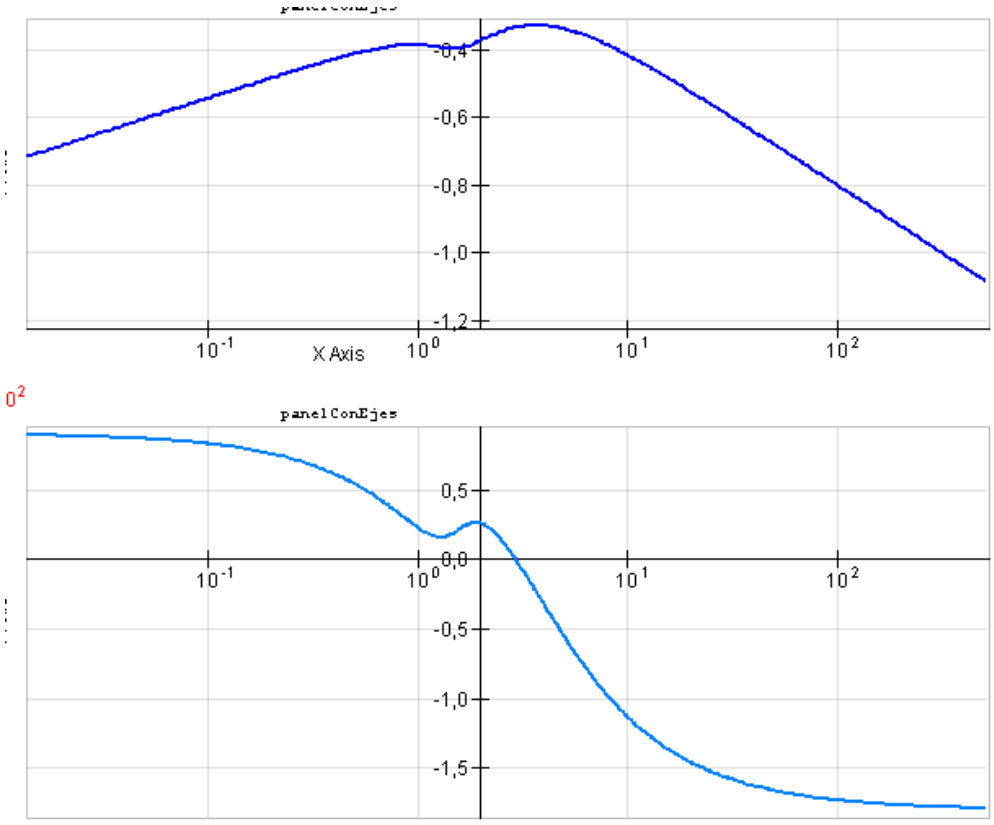


Fig. 17.4. Diagrama de Bode resultant amb EJS 3.

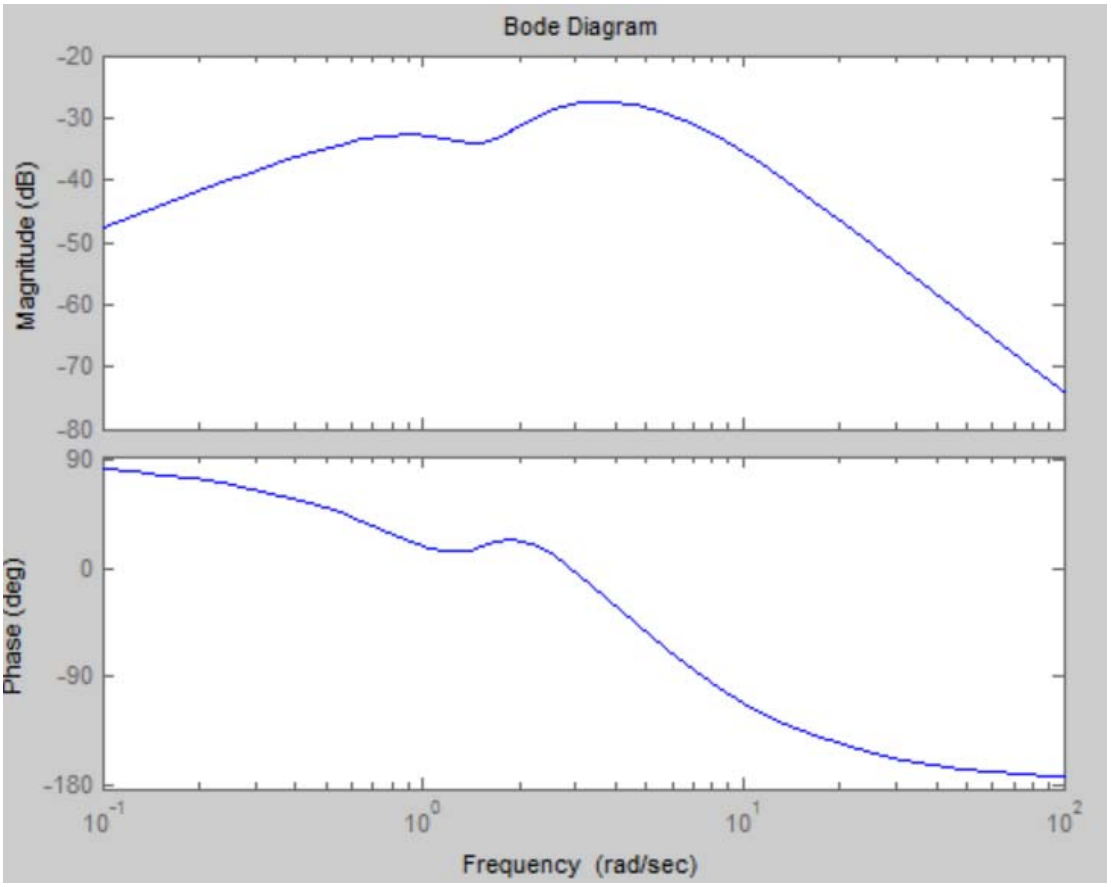


Fig. 17.5. Diagrama de Bode resultat amb MatLab 3.

E.8. Exemple pràctic de comparació amb un Bode de MatLab a la classe DTF.

Tot seguit es realitzarà un exemple de resultat obtingut mitjançant una aplicació creada en EJS en comparació amb el resultat obtingut de MatLab per la mateixa funció de transferència.

Fent servir la funció de transferència següent:

$$TF(z) = \frac{z + 0.2}{z^2 - 0.1z}$$

S'obté el següent a l'aplicació creada en EJS:

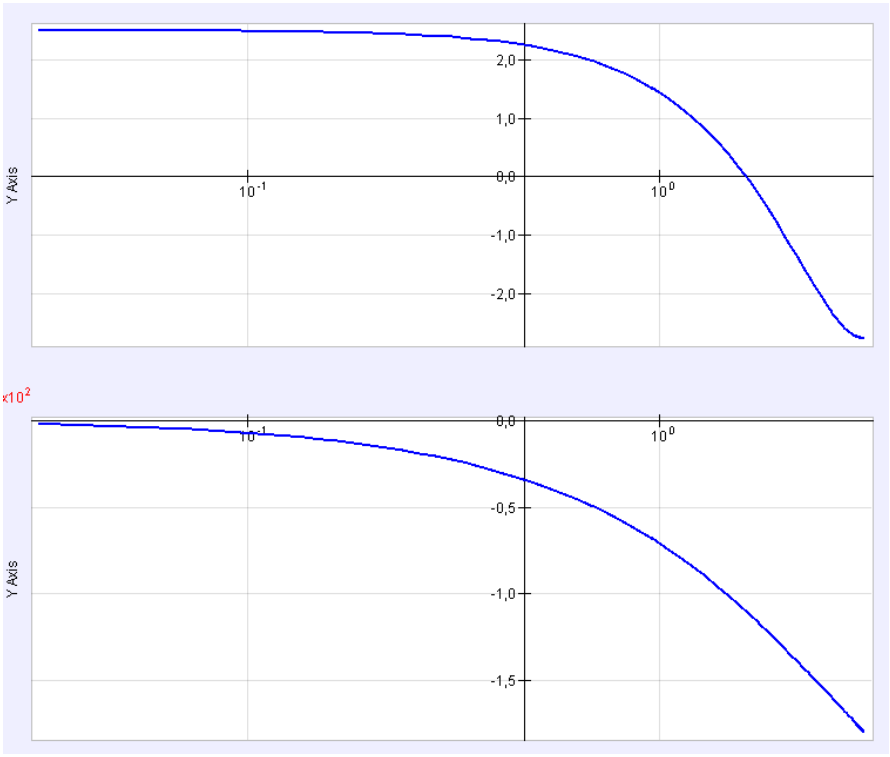


Fig. 17.7. Diagrama de Bode resultant amb EJS de la DTF.

Pel que fa a MatLab, s'obté el següent resultat:

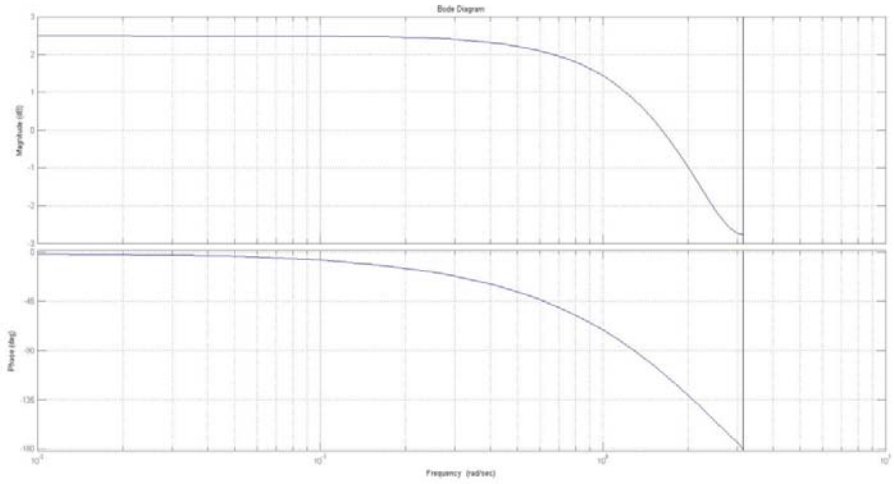


Fig. 17.8. Diagrama de Bode resultat amb MatLab de la DTF.

Fent servir la funció de transferència següent:

$$TF(z) = \frac{z(z-0.4)}{(z+0.2)(z+0.4+0.6j)(z+0.4-0.6j)(z\pm 0.5+0{,}5j)(z+0{,}5-0.5j)}$$

S'obté el següent a l'aplicació creada en EJS:

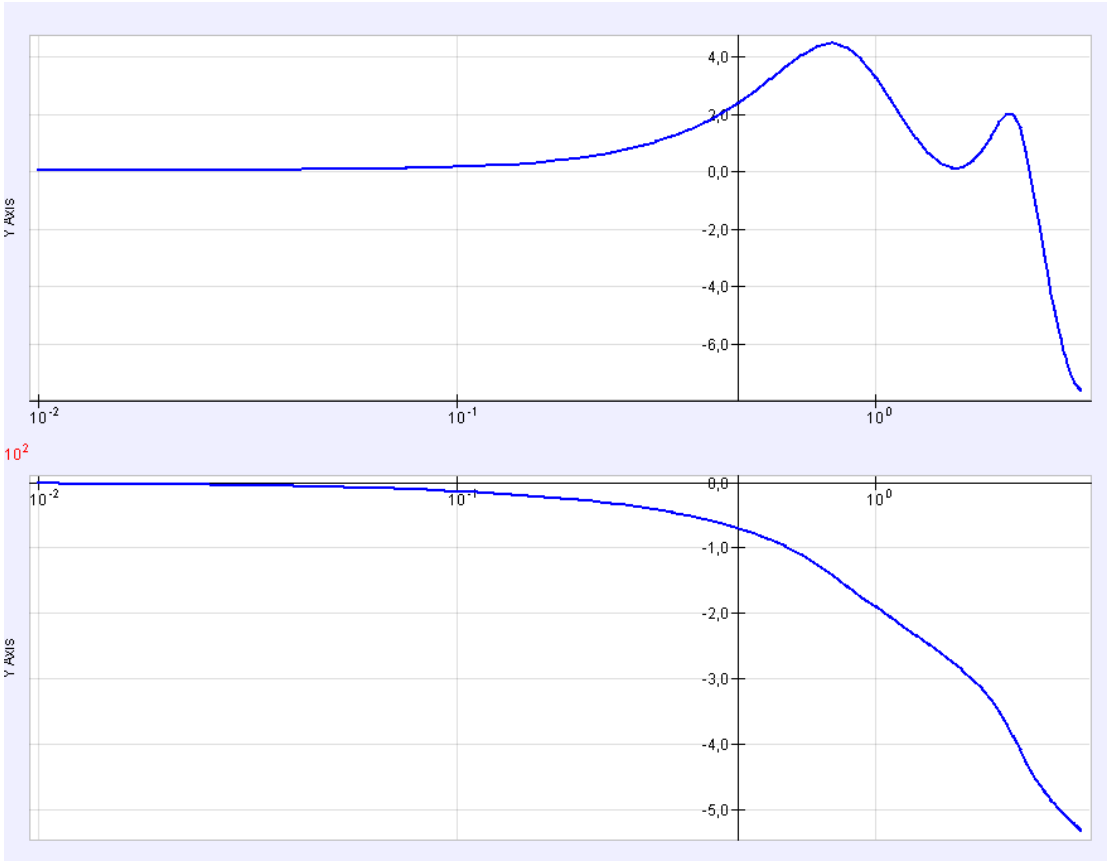


Fig. 17.7. Diagrama de Bode resultant amb EJS de la DTF.

Pel que fa a MatLab, s'obté el següent resultat:

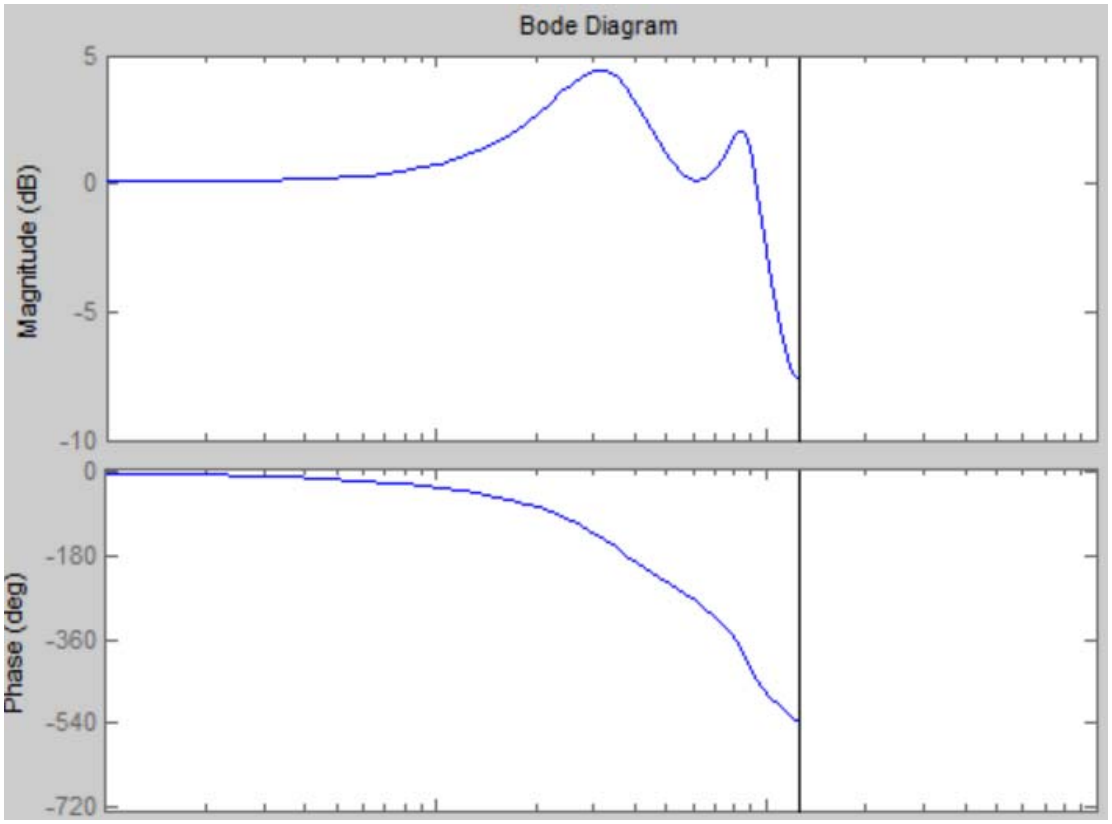


Fig. 17.8. Diagrama de Bode resultat amb MatLab de la DTF.

E.9. Exemple práctic del diagrama de Bode asimptòtic.

Tot seguit, es presenta un exemple de resultat de diagrama de Bode asimptòtic obtingut a partir de la llibreria Java creada i una aplicació feta amb EJS.

Fent servir el resultat de l'exemple del diagrama de Bode en una CTF, s'aprofitarà per traçar sobre aquest mateix diagrama en una aplicació EJS el diagrama de Bode asimptòtic:

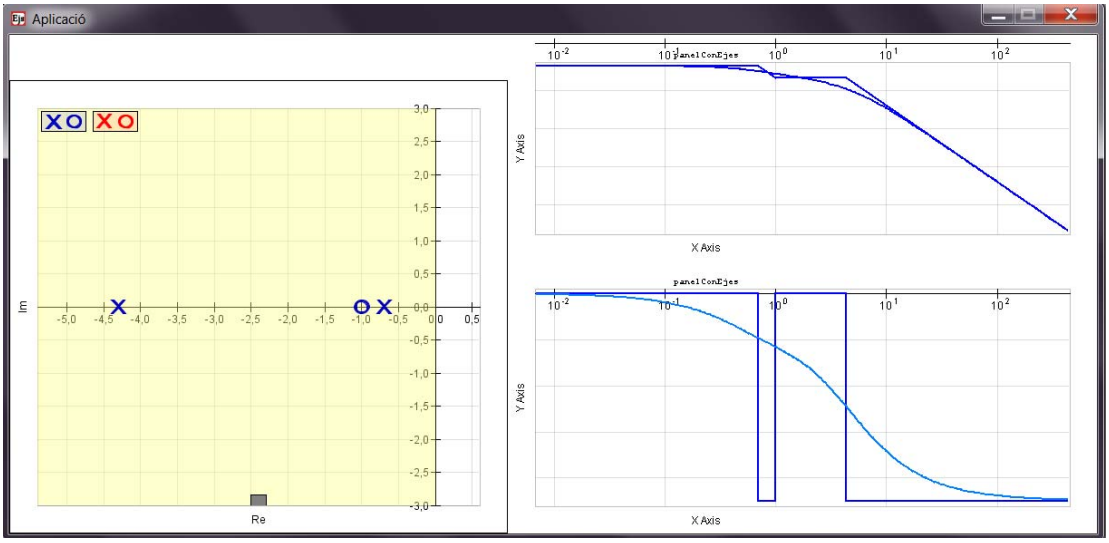


Fig. 18.2. Exemple d'aplicació on traça el diagrama de Bode asimptòtic.

Com es pot observar, el diagrama de Bode asimptòtic és molt coincident amb el diagrama de Bode real sobre tot en els punts on no hi ha canvis, que és sobretot on els diagrames han de coincidir.

Un segon exemple és el següent:

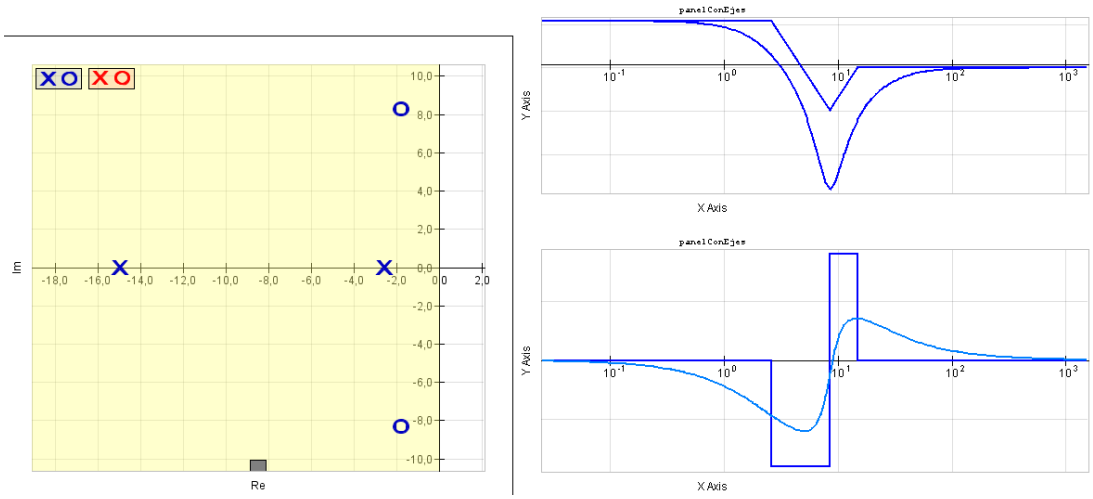


Fig. 18.2. Exemple d'aplicació on traça el diagrama de Bode asimptòtic.

Un tercer exemple és el següent:

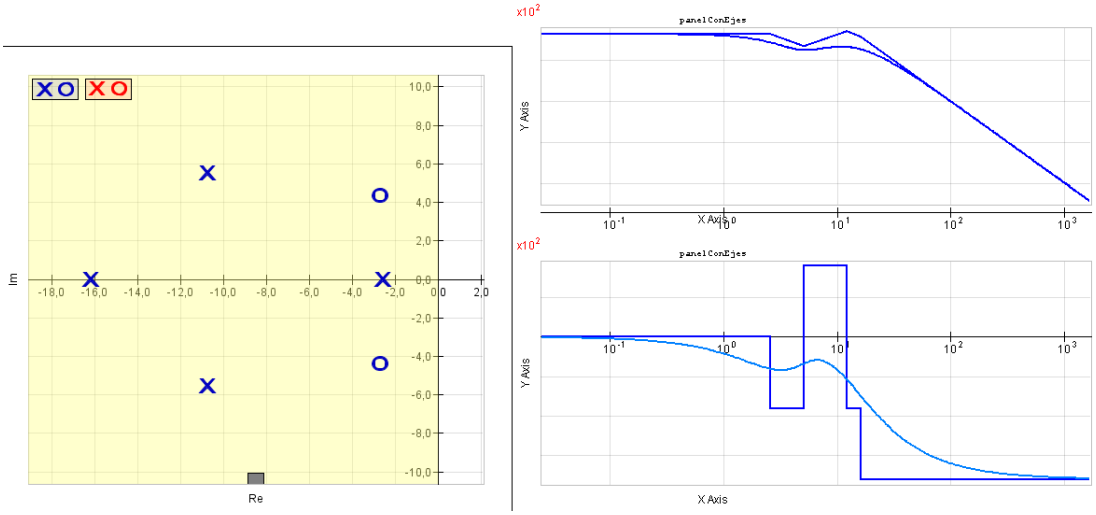


Fig. 18.2. Exemple d'aplicació on traça el diagrama de Bode asimptòtic.

E.10. Exemple práctic del diagrama de Nyquist en CTF

En aquest cas es farà servir la funció de transferència següent:

$$TF(s) = \frac{2s^2 + 8}{s^2 + 8s + 16} = 2 \cdot \frac{(s + 2{,}000j)(s - 2{,}000j)}{(s + 4)(s + 4)}$$

A l'aplicació creada expressament de l'EJS, s'obté el següent resultat:

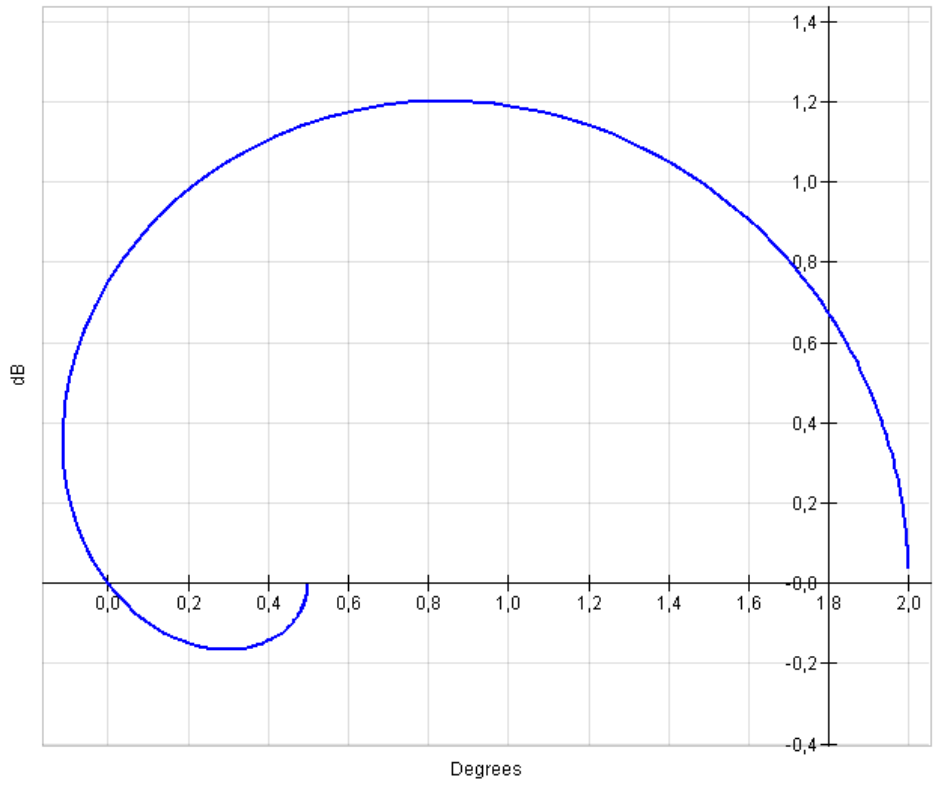


Fig. 19.2. Resultat obtingut en una aplicació EJS que representa el diagrama de Nyquist.

Pel que fa al resultat obtingut a MatLab de la mateixa funció de transferència tenim:

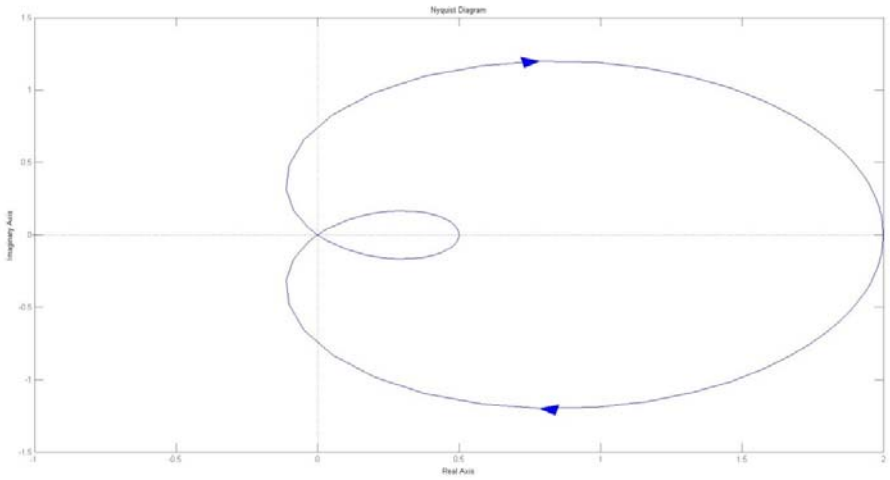


Fig. 19.3. Resultat obtingut en MatLab que representa el diagrama de Nyquist.

E.11. Exemple práctic del diagrama de Nyquist en DTF

Tot seguit es passarà a realitzar el mateix pas anterior, però en aquest cas amb una DTF.

En aquest cas es farà servir la funció de transferència següent:

$$TF(z) = \frac{z + 0.75}{z^2 - 0.25} = \frac{z + 0.75}{(z + 0.5)(z - 0.5)}$$

A l'aplicació creada expressament de l'EJS, s'obté el següent resultat:

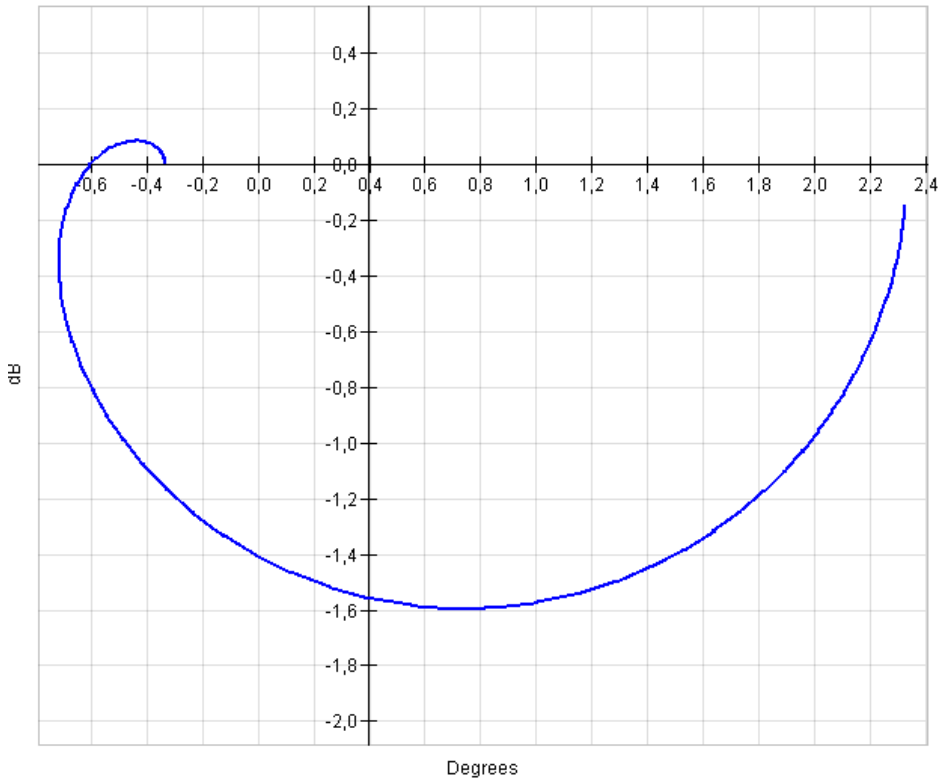


Fig. 19.4. Resultat obtingut en una aplicació EJS que representa el diagrama de Nyquist.

Pel que fa al resultat obtingut a MatLab de la mateixa funció de transferència tenim:

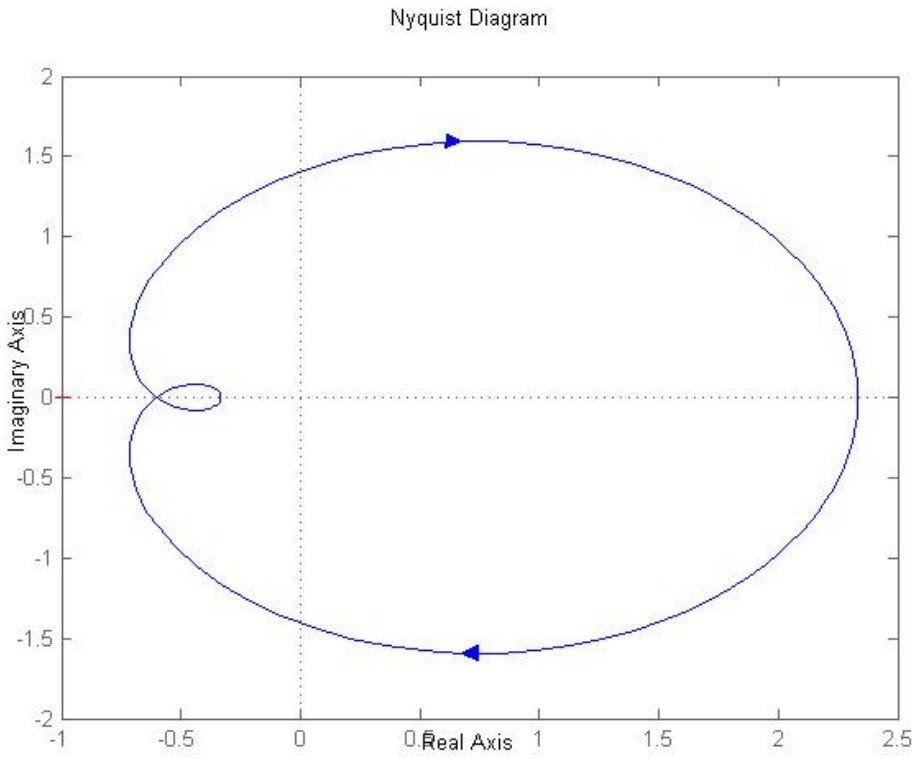


Fig. 19.5. Resultat obtingut en MatLab que representa el diagrama de Nyquist per la DTF.

Com en el cas del temps continu, en aquesta aplicació de la DTF no s'ha aplicat la simetria, però els resultats són correctes.

E.12. Exemple práctic del diagrama de Nichols en CTF

Tot seguit es passa a fer un exemple pràctic de demostració del diagrama de Nichols amb la següent funció de transferència:

$$TF(s) = \frac{2s^2 + 4s + 8}{s^2 + 10s + 2} = 2 \cdot \frac{(s + 1 + 1{,}732j)(s + 1 - 1{,}732j)}{(s + 0.204)(s + 9{,}795)}$$

El resultat obtingut mitjançant una aplicació EJS creada expressament ha estat:

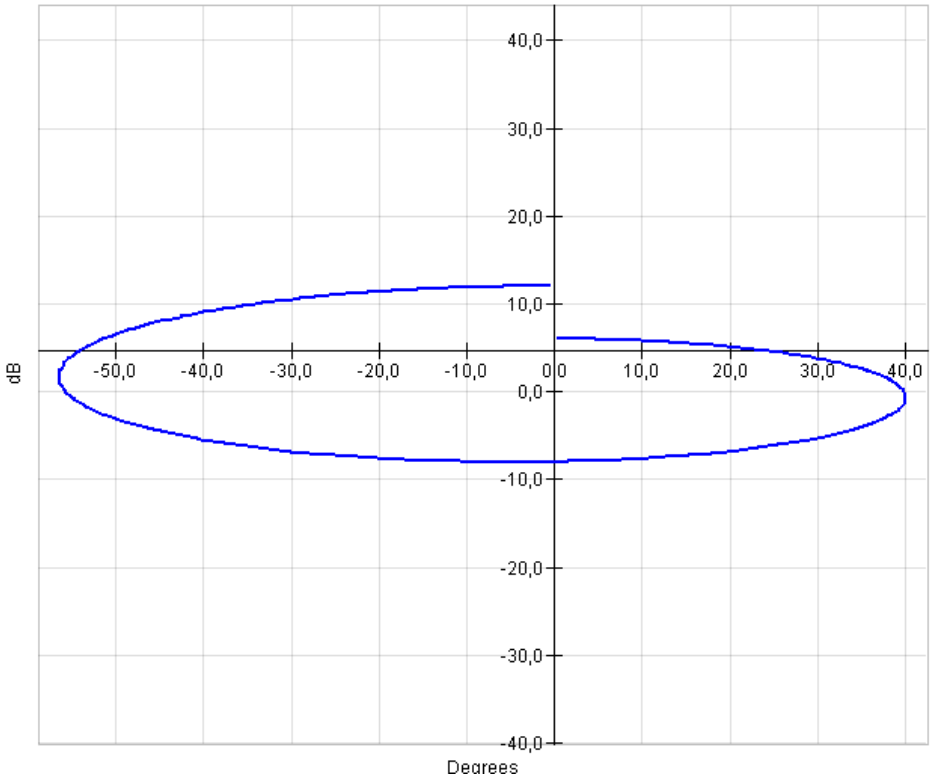


Fig. 20.2. Diagrama de Nichols obtingut mitjançant una aplicació EJS i la llibreria.

El resultat obtingut mitjançant MatLab ha estat:

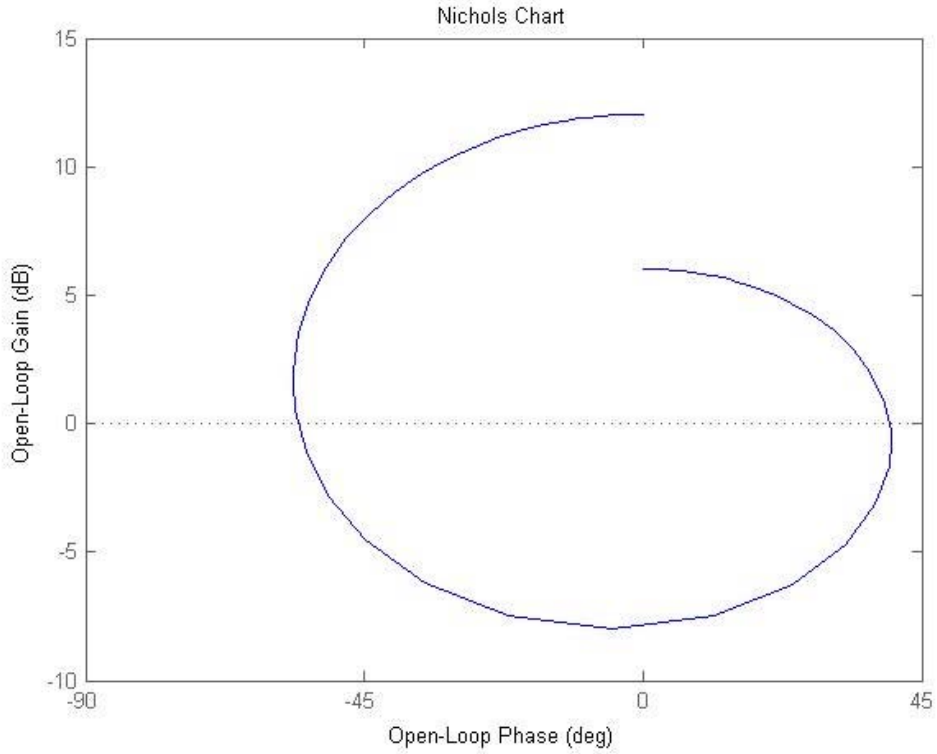


Fig. 20.3. Diagrama de Nichols obtingut mitjançant MatLab.

Com en tots els casos anteriors, els diagrames obtinguts són molt semblants a simple vista i tenen una representació molt igual.

E.13. Exemple práctic del diagrama de Nichols en DTF

Tot seguit es passa a fer els mateixos passos que el cas anterior amb aquest cas però, amb una DTF:

$$TF(Z) = \frac{4}{z - 0.24}$$

El resultat obtingut mitjançant una aplicació EJS creada expressament ha estat:

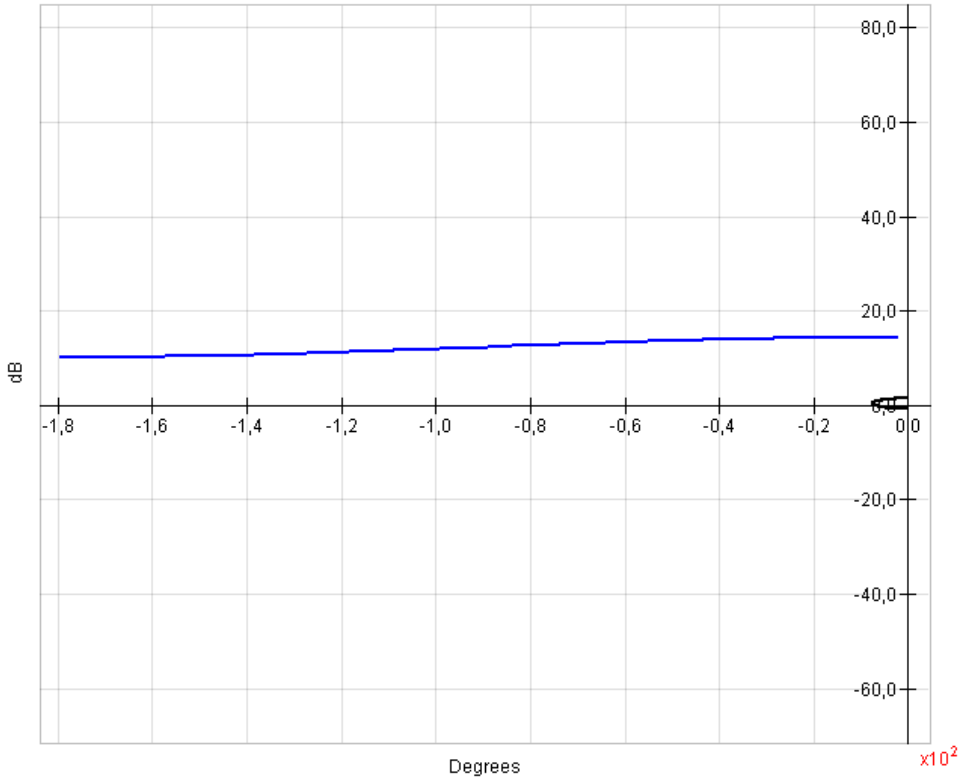


Fig. 20.4. Diagrama de Nichols obtingut mitjançant una aplicació EJS i la llibreria.

El resultat obtingut mitjançant MatLab ha estat:

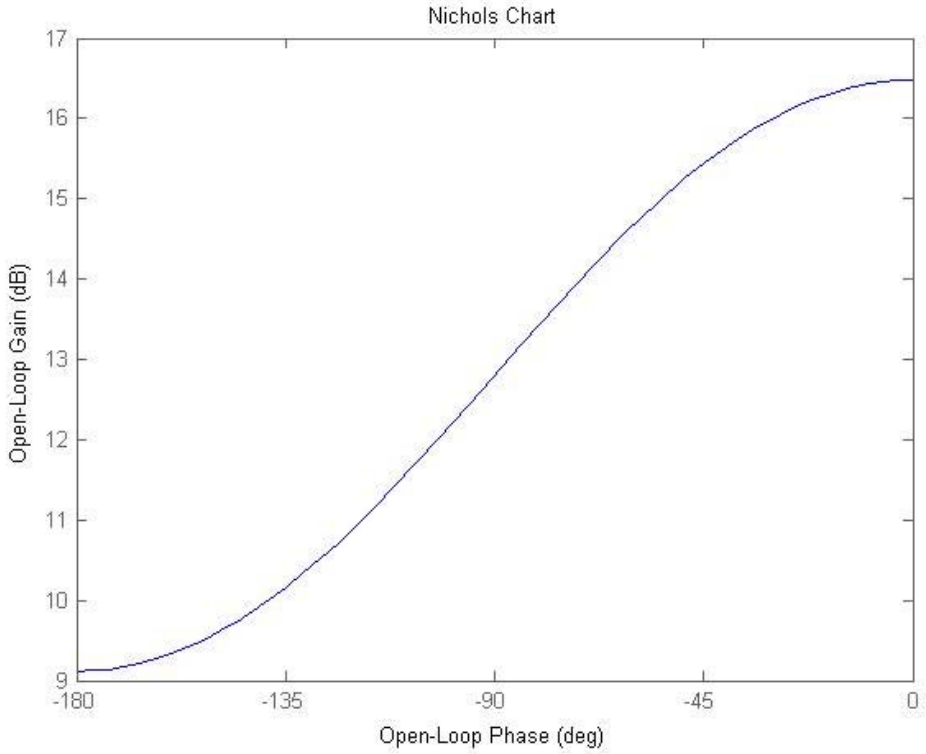


Fig. 20.5. Diagrama de Nichols obtingut mitjançant MatLab.

E.14. Exemple práctic d'una resposta temporal en CTF

Tot seguit es passarà a comparar la resposta temporal obtinguda mitjançant una aplicació creada expressament en EJS i una resposta obtinguda a partir de MatLab per a un objecte de la classe CTF.

En aquest cas, es farà servir la funció de transferència d'exemples anteriors:

$$X(s) = \frac{s-1}{s^3 + 2s^2 + s}$$

La resposta temporal obtinguda a part de l'aplicació EJS és la següent:

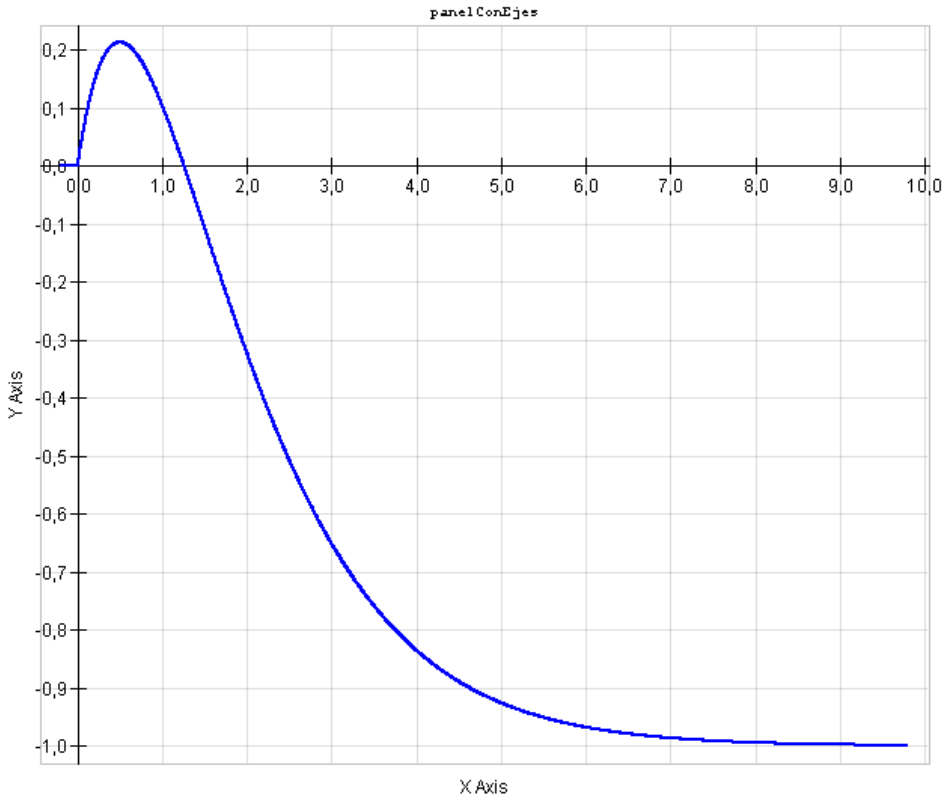


Fig. 21.2. Resposta temporal obtinguda mitjançant EJS i la llibreria Java creada CTF.

Fent servir la mateixa funció de transferència, la resposta temporal obtinguda a MatLab Simulink i l'esquema que s'ha fet servir són els següents:

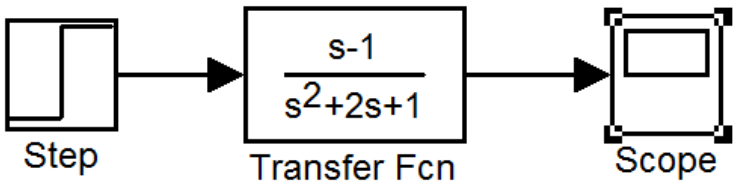


Fig. 21.3. Esquema de MatLab Simulink per obtenir la resposta temporal pel temps continu.

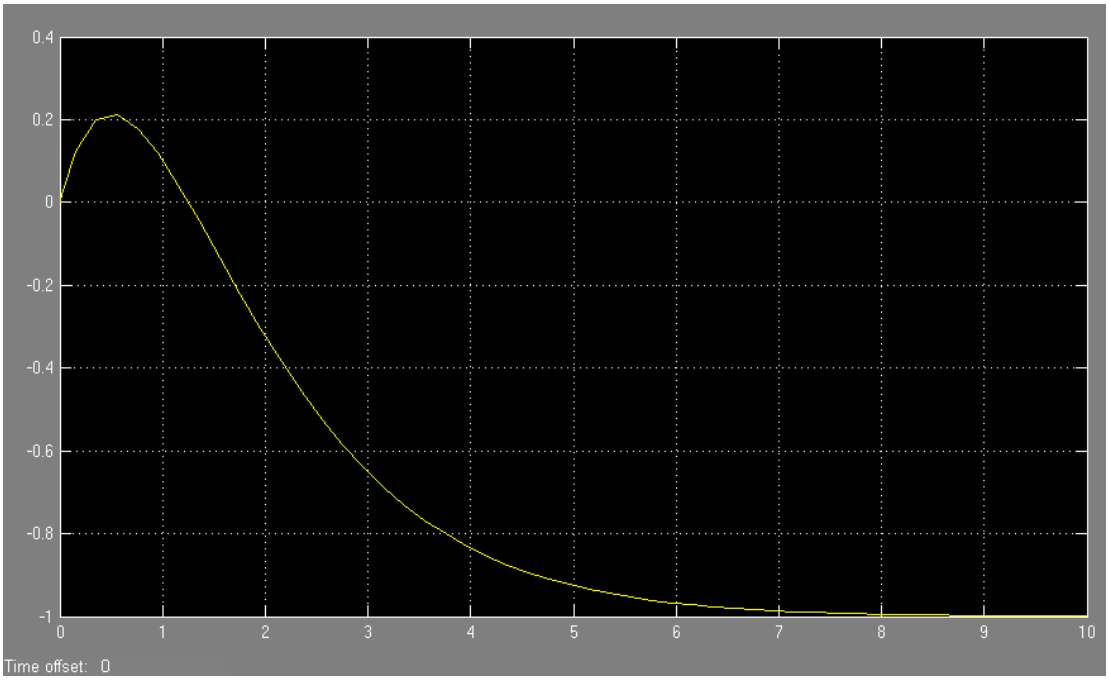


Fig. 21.4. Resposta temporal obtinguda mitjançant MatLab Simulink en temps continu.

Les dos representacions obtingudes són molt semblants. A les futures aplicacions creades en EJS mitjançant la llibreria, se'ls permet la flexibilitat de fer representacions diferents i ampliacions del gràfic obtingut etc. Tot mitjançant les definicions que es realitzin en EJS (les quals surten de l'objecte d'aquest projecte).

E.15. Exemple práctic d'una resposta temporal en DTF

Tot seguit es passarà a comparar la resposta temporal obtinguda mitjançant una aplicació creada expressament en EJS i una resposta obtinguda a partir de MatLab per a un objecte de la classe DTF. En aquest cas, es farà servir la resposta següent (considerant una entrada graó):

$$X(s) = \frac{z}{z-1} \cdot \frac{z-0.5}{z^2+z+0.25}$$

La resposta temporal obtinguda a part de l'aplicació EJS és la següent:

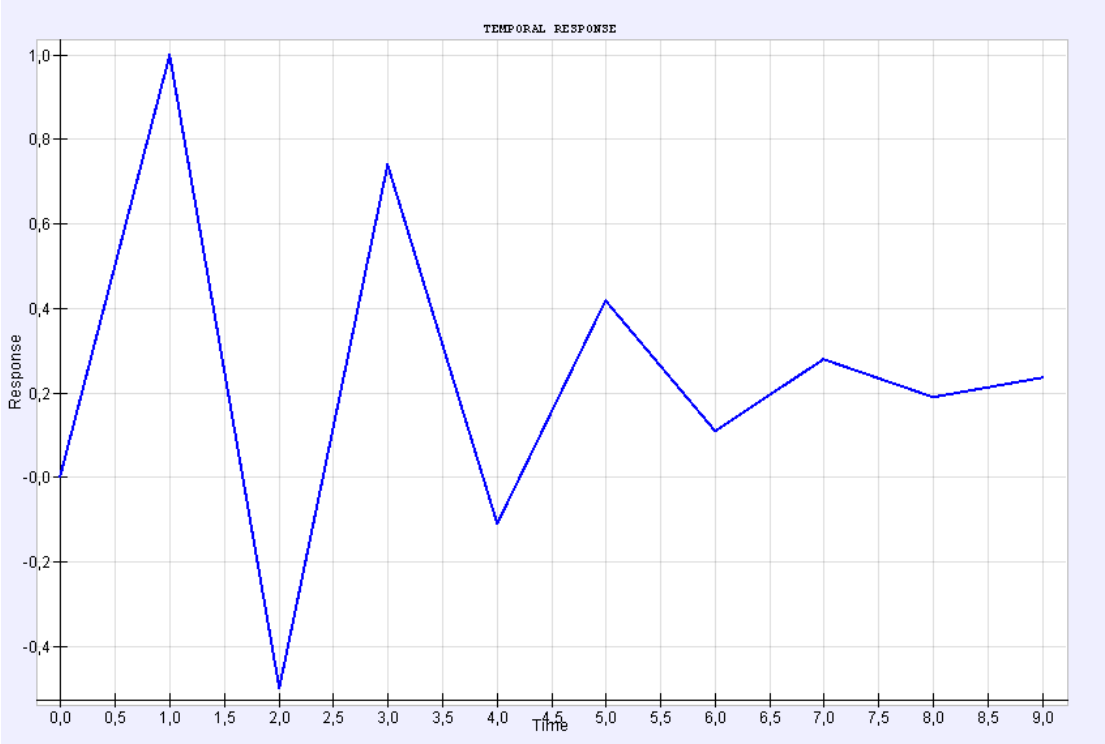


Fig. 21.5. Resposta temporal obtinguda mitjançant EJS i la llibreria Java creada DTF.

Fent servir la mateixa funció de transferència, la resposta temporal obtinguda a MatLab Simulink i l'esquema que s'ha fet servir són els següents:

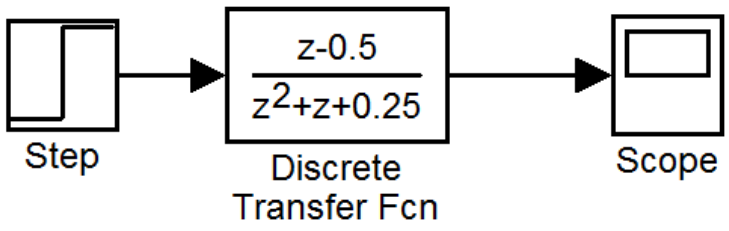


Fig. 21.6. Esquema de MatLab Simulink per obtenir la resposta temporal en temps discret.

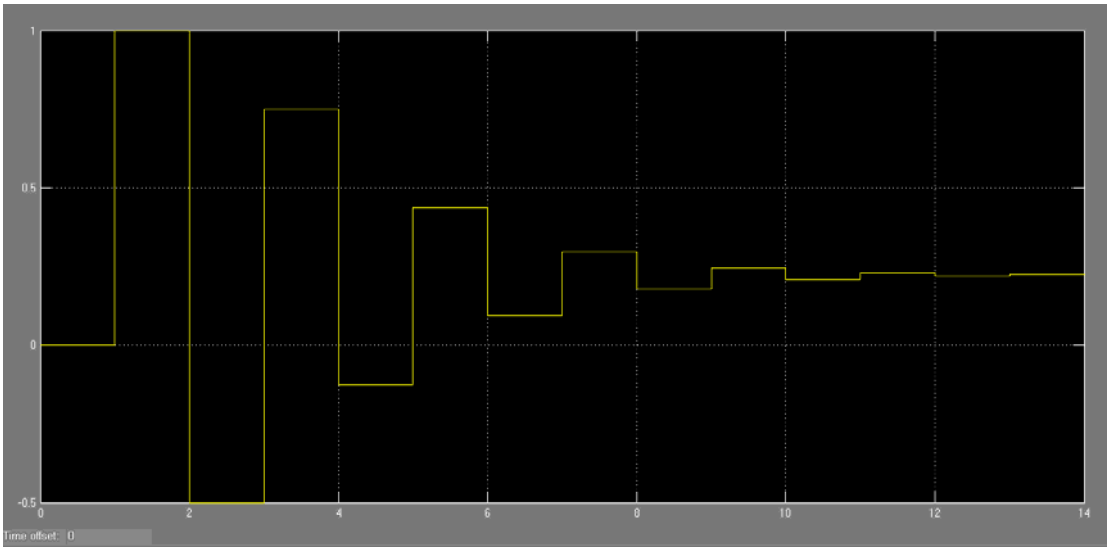


Fig. 21.7. Resposta temporal obtinguda mitjançant MatLab Simulink en temps discret.

Les dos representacions obtingudes són molt semblants. A les futures aplicacions creades en EJS mitjançant la llibreria, se'ls permet la flexibilitat de fer representacions diferents i ampliacions del gràfic obtingut etc. Tot mitjançant les definicions que es realitzin en EJS (les quals surten de l'objecte d'aquest projecte).

F. Codis de test de MatLab

El següent capítol montra els codis fets servir i grafics obtinguts per tal de realitzar una comprovació dels resultat obtinguts a partir de la llibreria creada en el projecte i els resultats obtinguts en les mateixes operacions en MatLab.

F.1. Creació del diagrama de Bode per la comparació

F.1.1. Bode a CTF

trans = tf([2 2],[1 5 3])

bode(trans)

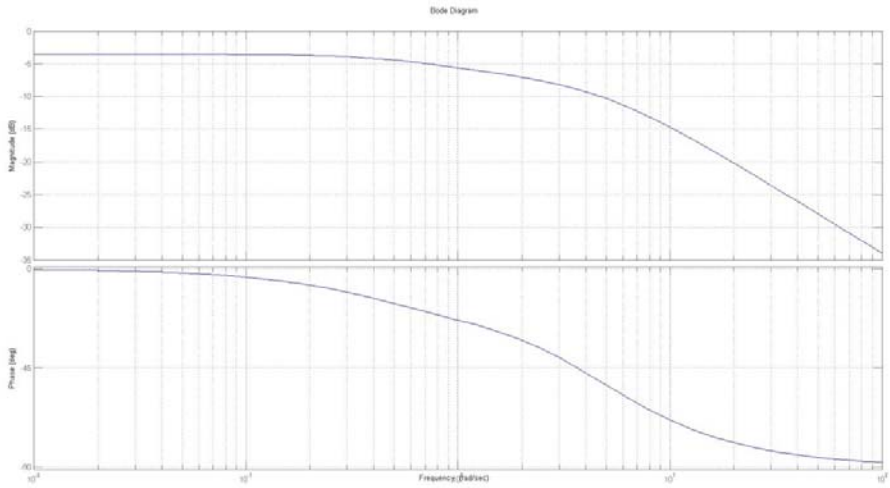


Fig. E.1. Diagrama de Bode creat a mitjançant MatLab per a una comparació amb CTF.

trans = tf([1 -0.5],[1 4])

bode(trans)

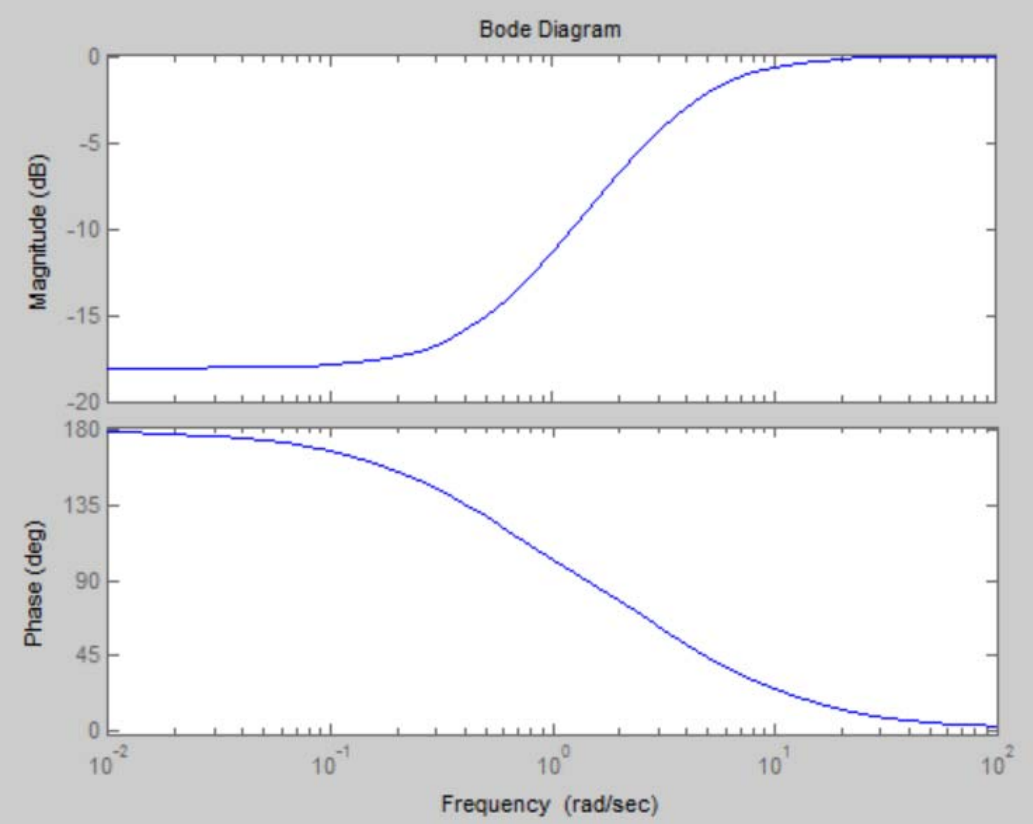


Fig. E.1. Diagrama de Bode creat a mitjançant MatLab per a una comparació amb CTF.

trans = tf([2 2 5 0],[1 13 70 172.51 221.31 119.53])

bode(trans)

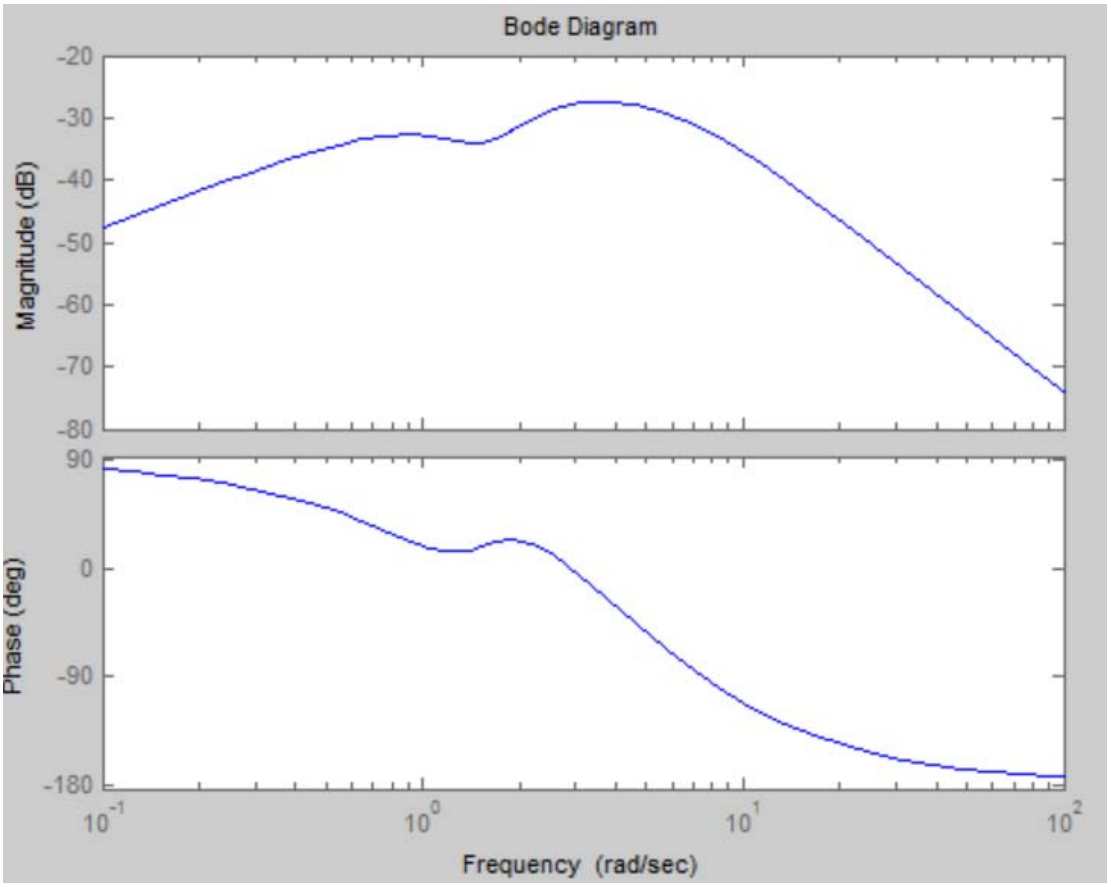


Fig. E.1. Diagrama de Bode creat a mitjançant MatLab per a una comparació amb CTF.

F.1.2. Bode a DTF

```
tf = filt([0 1 0.2],[1 -0.1])

bode(tf)
```

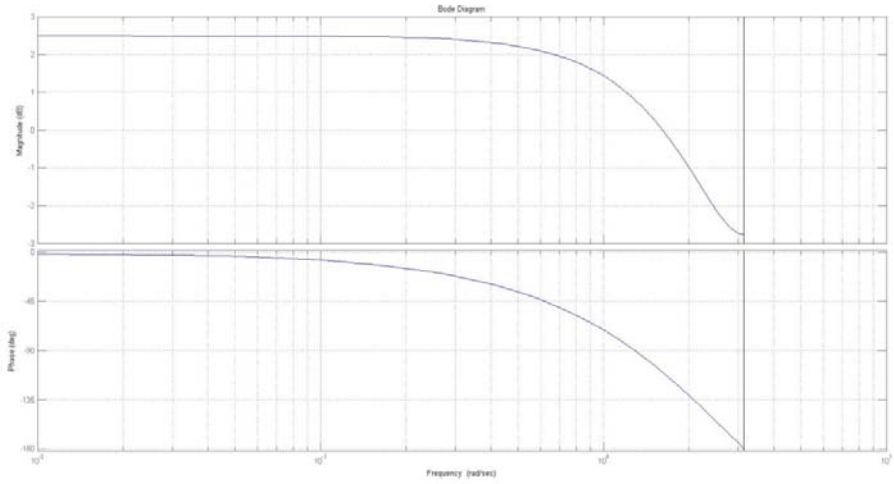
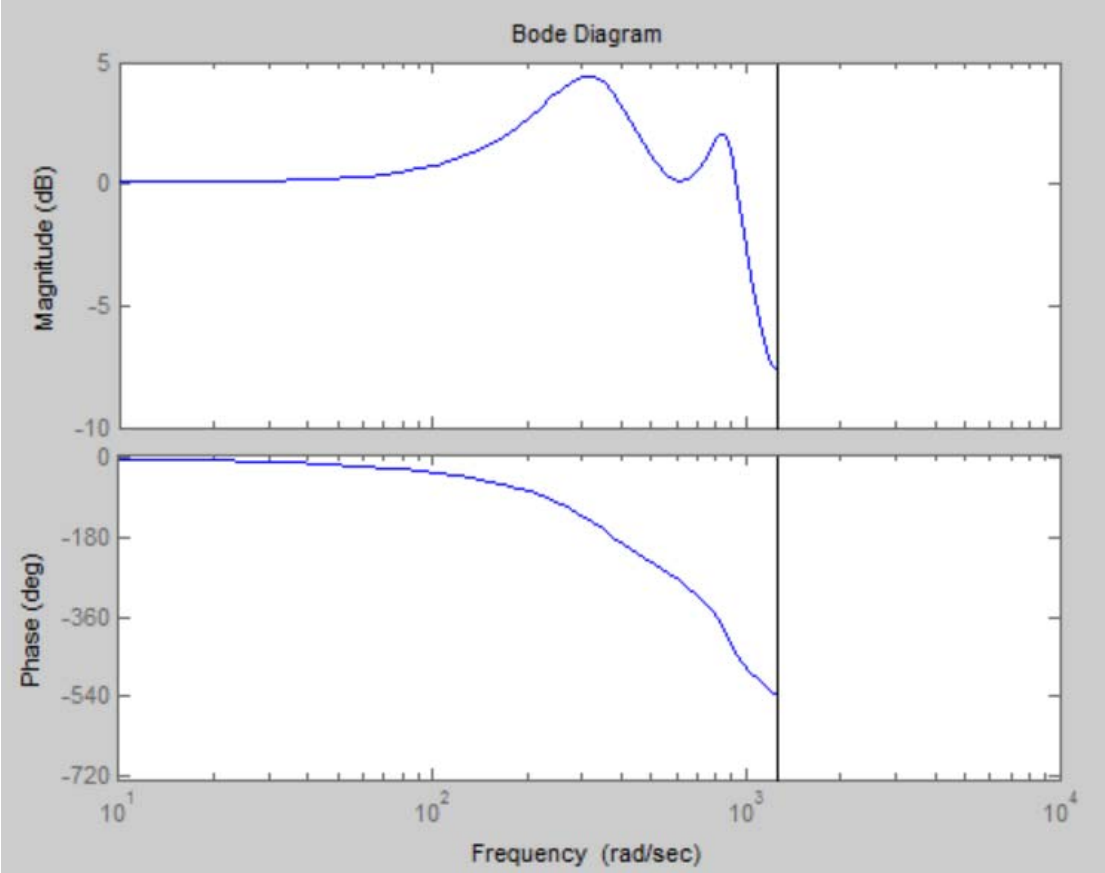


Fig. E.2. Diagrama de Bode creat a mitjançant MatLab per a una comparació amb DTF.

```
>> tf1 = tf([1 0.4 0],[1 0 0.18 -0.08 0.24 0.05],0.0025);

>> bode(tf1)
```



F.2. Creació del diagrama de Nichols per la comparació

F.2.1. Nichols CTF

trans = tf([2 4 8],[1 10 2])

nichols(trans)

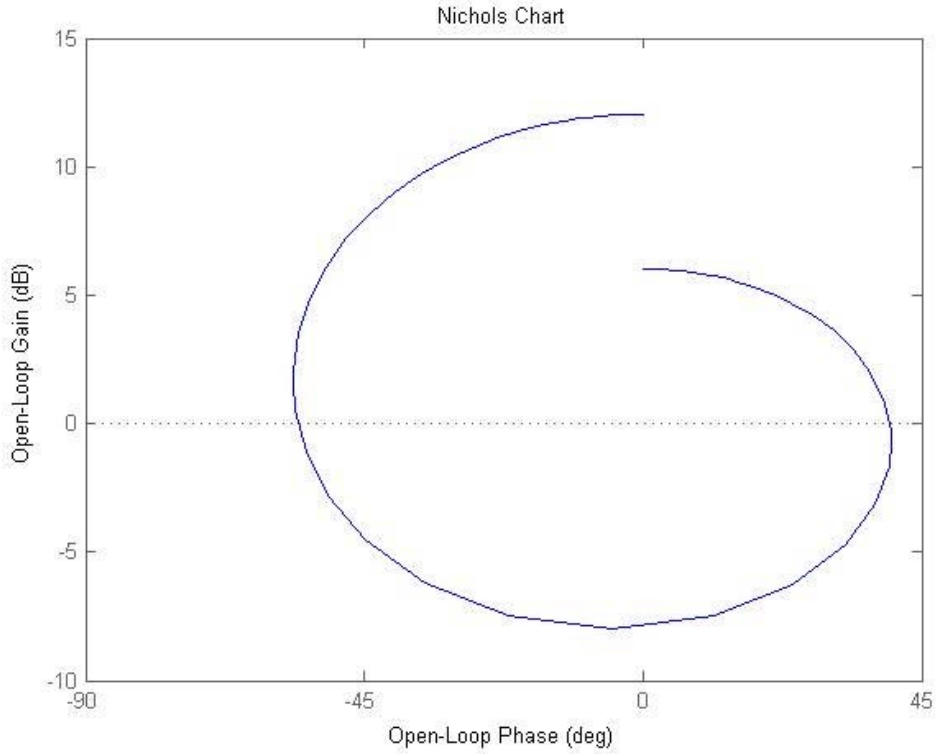


Fig. E.3. Diagrama de Nichols creat mitjançant MatLab per a una comparació amb CTF.

F.2.2. Nichols DTF

trans=filt([0 4],[1 -0.4])

nichols(trans)

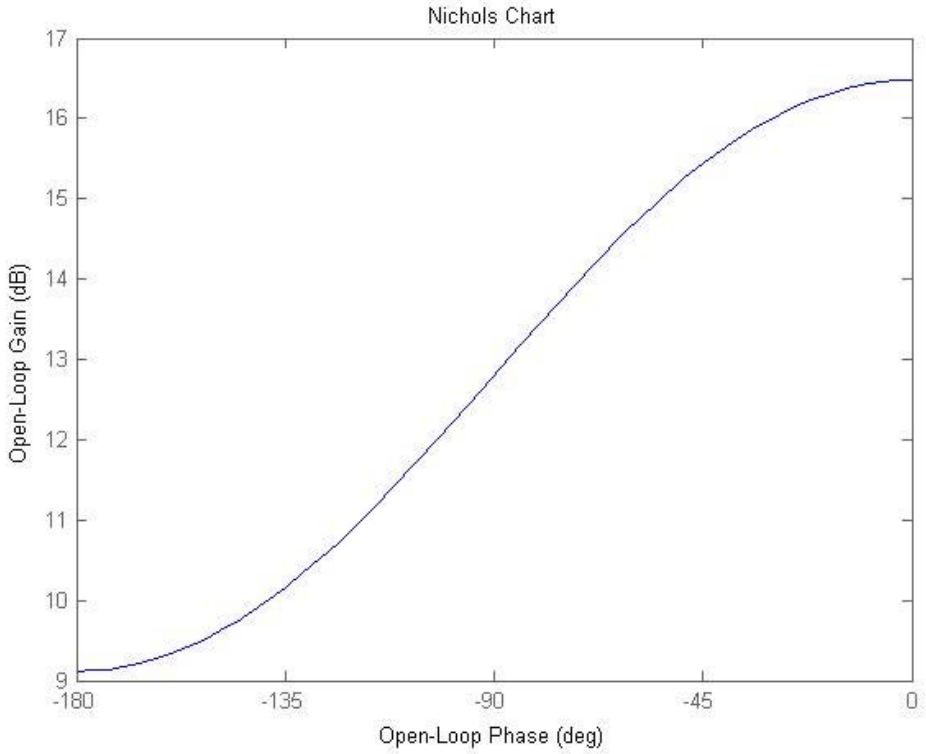


Fig. E.4. Diagrama de Nichols creat mitjançant MatLab per a una comparació amb DTF.

F.3. Creació del diagrama de Niquist per la comparació

F.3.1. Nyquist CTF

```
trans = tf([2 0 8],[1 8 16])

nyquist(trans)
```

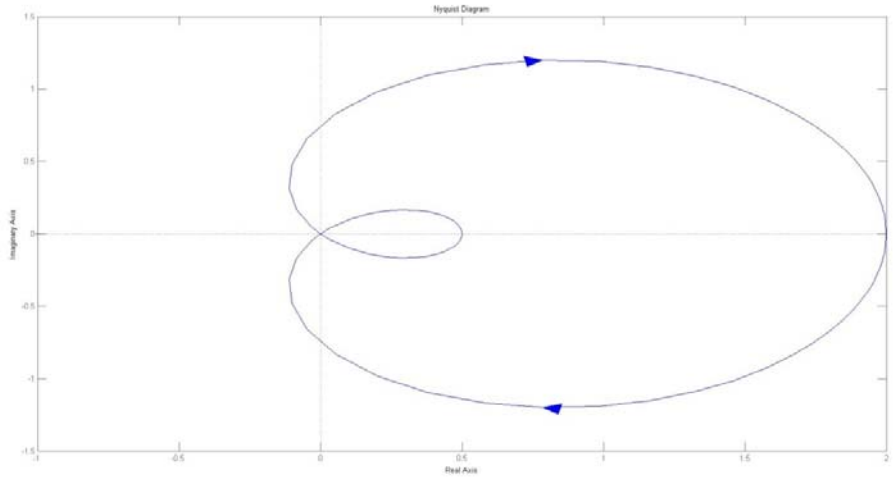


Fig. E.5. Diagrama de Nyquist creat mitjançant MatLab per a la comparació amb una CTF.

F.3.2. Nyquist DTF

trans = filt([0 1 +0.75],[1 0 -0.25])

nyquist(trans)

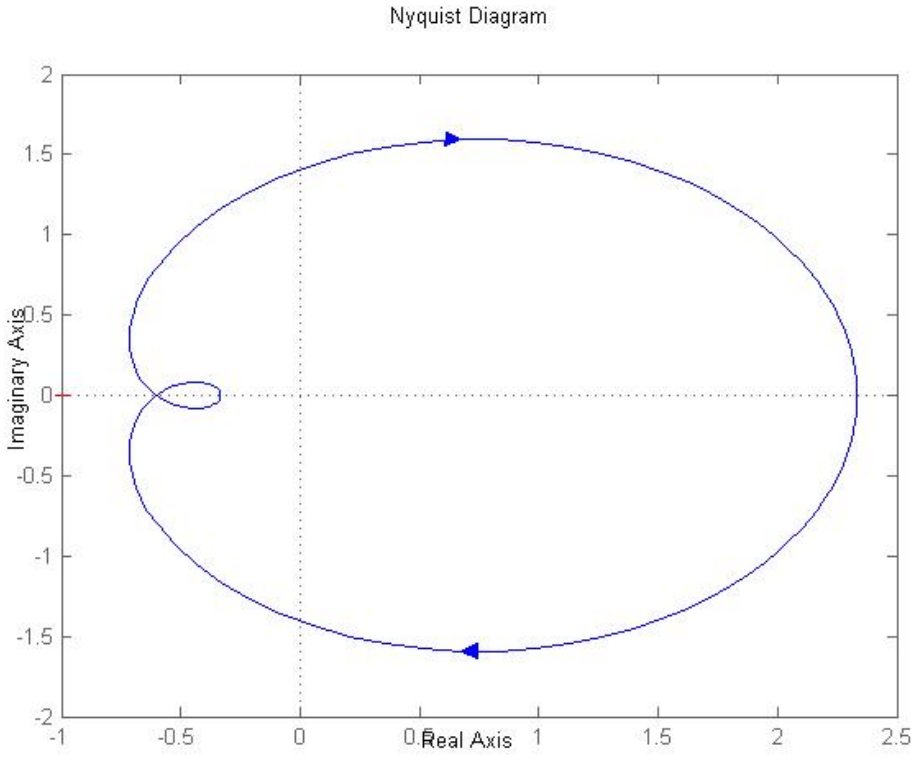


Fig. E.6. Diagrama de Nyquist creat mitjançant MatLab per a la comparació amb una DTF.

F.4. Creació d'una resposta temporal per la comparació

Per a la visualització de la resposta temporal, s'han creat dos models de MatLab Simulink, una per la CTF i una altre per la DTF.

F.4.1. Model de Simulink per resposta temporal en temps continu

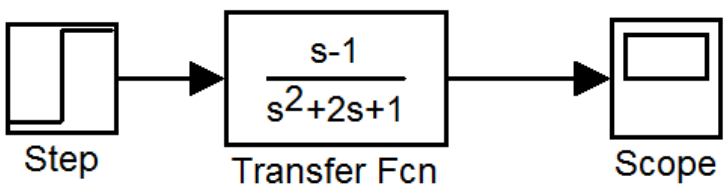


Fig. E.7. Model de Simulink per representar la resposta temporal en temps continu.

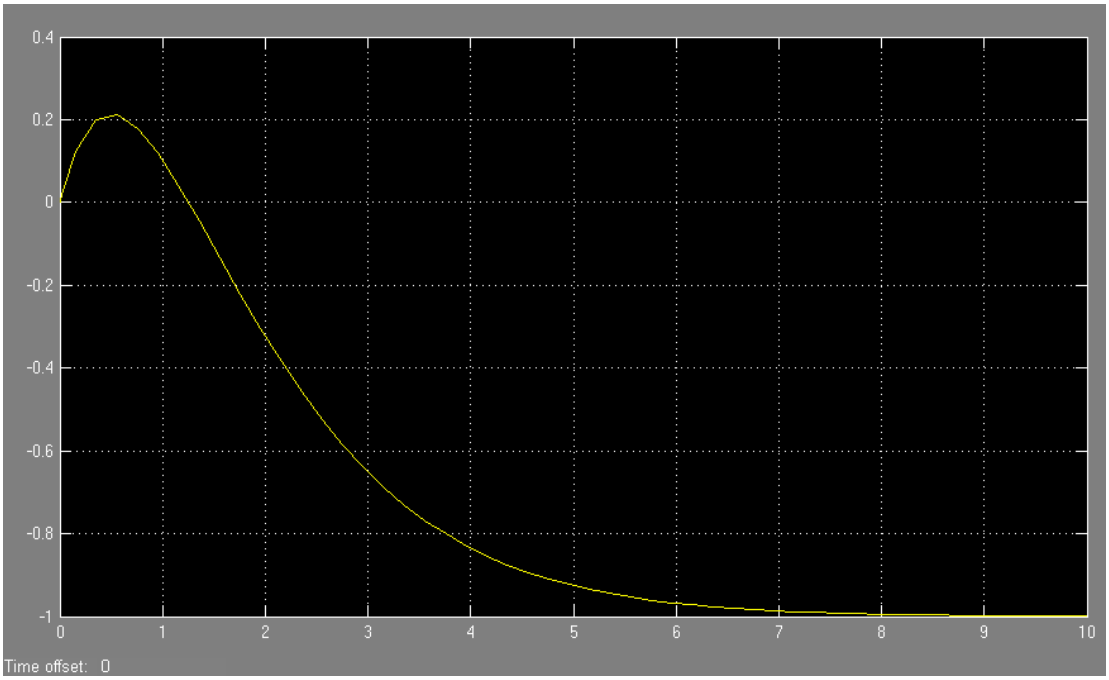


Fig. E.8. Resposta temporal en temps continu.

F.4.2. Model de Simulink per resposta temporal en temps discret.

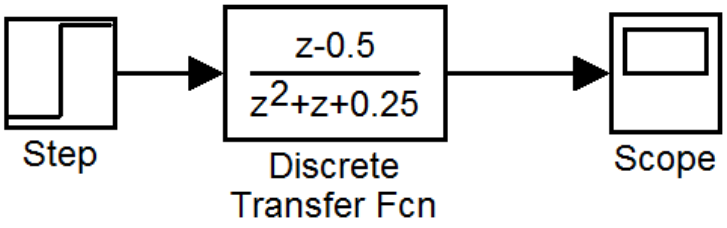


Fig. E.9. Model de Simulink per representar la resposta temporal en temps discret.

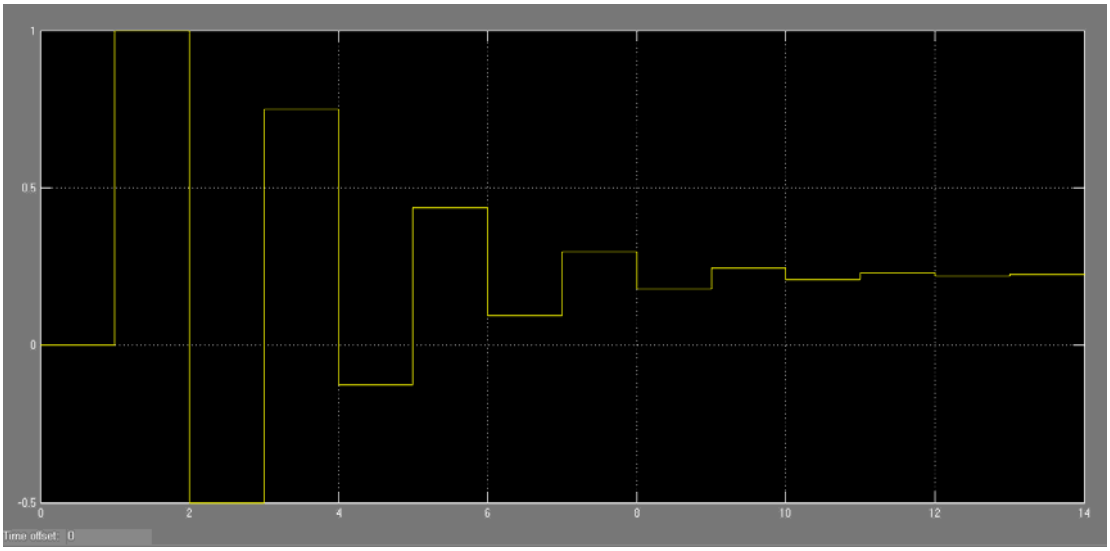


Fig. E.10. Resposta temporal en temps discret.

G. Mètodes auxiliars de les classes

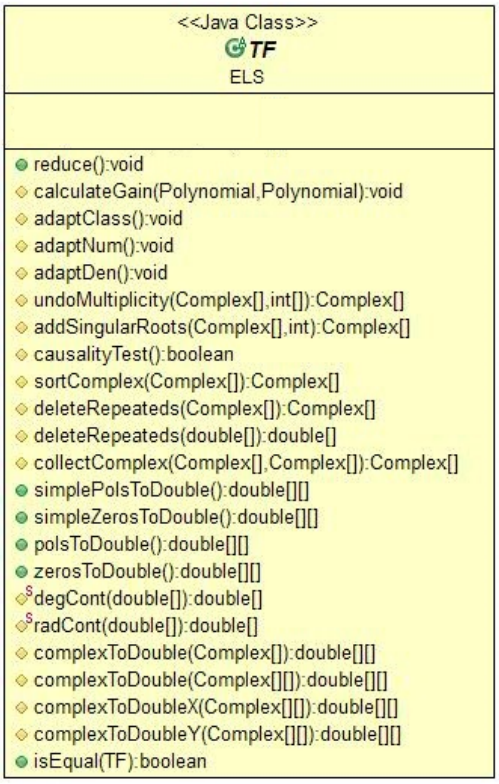


Fig. G.1. Mètodes auxiliar de la classe TF.

La classe TF disposa internament d'un seguit de mètodes per tal de funcionar correctament, fer comprovacións, modificar representacions i realitzar adaptacions cap a EJS. Aquests mètodes, es tracten en general de mètodes privats, es a dir, que poden ser utilitzats per les seves classes derivades, existint alguna excepció pública, que pot esser utilitzada externament.

G.1.1. Reducció de dades

El primer mètode de tots, el mètode *reduce* es tracta d'un mètode que serveix per fer reducció de dades de l'estructura de la TF. Aquesta reducció es tracta, per exemple, de eliminar arrels del numerador, que també es trobin amb el mateix valor en el denominador. O també eliminar integradors amb derivadors.

Per posar un exemple, se suposa que amb un constructor es crida la següent funció de transferència:

$$TF(s) = 4 \cdot \frac{(s-2)^2 \cdot (s-4)^1 \cdot s^2}{(s-1) \cdot (s-2) \cdot s^4} \cdot e^{-s6}$$

Fent la crida al mètode *reduce* s'acabarà obtenint la següent reducció:

$$TF(s) = 4 \cdot \frac{(s-2) \cdot (s-4)^1}{(s-1) \cdot s^2} \cdot e^{-s6}$$

Ocupant menys espai en memòria i alleugerant molts càlculs.

Aquest mètode s'ha decidit implementar-lo de forma que s'hagi de fer un crida externa degut a que si es realitzés de forma interna, es realitzarien reduccions eliminant dades que podria no interessar eliminar i per tant, es deixa triar a l'usuari quan fer aquestes reduccions.

G.1.2. Càlcul de guany en constructors

El següent mètode que es troba a la llista es tracta del mètode *calculateGain* aquest mètode serveix per extreure el valor de guany k en cas de que s'hagi fet la crida d'un constructor entrant com a paràmetres d'entrada polinomis per definir el numerador i el denominador.

Matemàticament, realitzaria la següent operació internament a l'estructura de la TF:

$$TF(s) = \frac{4s^4 + 8s + 16}{2s^6 + 6s^2 + 4} \rightarrow \frac{4}{2} \cdot \frac{s^4 + 2s + 4}{s^6 + 3s^2 + 2} \rightarrow 2\frac{s^4 + 2s + 4}{s^6 + 3s^2 + 2}$$

G.1.3. Adaptació a l'estructura interna

El mètodes *adapt* que es troben en tercer lloc de la llista, es tracten de mètodes que serveixen per, una vegada definits quins paràmetres d'entrada té el constructor, adaptar les dades a l'estructura interna de la classe.

Per posar un exemple, si el constructor cridat se li donen com a paràmetres d'entrada dos polinomis (numerador i denominador) i un retard, el mètode *adaptClass* realitza la adaptació d'aquestes dades a l'estructura interna explicada a l'inici d'aquest capítol.

G.1.4. Preparació per la realització de càlculs

Existeixen un conjunt de mètodes que serveixen per poder realitzar càlculs i obtenir determinats valors a partir de l'estructura interna de la classe. Només llegint el nom dels mètodes es pot deduir les tasques que realitzen. Per aclarir-ho més es farà un petit incís.

En primer lloc es troba el mètode *undoMultiplicity*, aquest mètode, donat un vector de complexes que representen les arrels del sistema i un vector de la mateixa longitud d'enters que representa els exponents pels quals estan elevades aquestes arrels, retorna un vector de complexes que representen totes les arrels:

$$\begin{matrix}\{s,(s-2),(s-3)\}\\ \{3,2,1\}\end{matrix}\Big|\rightarrow \quad \{s,s,s,(s-2),(s-2),(s-3)\}$$

El segon mètode *addSingularRoots* afegeix al vector d'arrels donat com a paràmetre d'entrada, el nombre d'arrels singulars indicat com a segon paràmetre d'entrada:

$$\begin{matrix}\{(s-2),(s-3)\}\\ 4\end{matrix}\Big|\rightarrow \quad \{s,s,s,s,(s-2),(s-3)\}$$

El quart mètode que es troba és el *sortComplex* aquest mètode ordena un vector de complexes tot seguint els següents criteris:

1. Ordena de valor absolut més petit a valor absolut més gran.
2. En cas de tenir el mateix valor absolut, posa primer el complex que té la part real més gran.
3. Si en els dos casos anteriors hi ha el mateix valor, posa primer el que té la part imaginaria més petita.

El cinquè mètode per preparar l'estructura a realitzar càlculs és el mètode *deleteRepeateds*. Aquest mètode elimina els valors repetits de dins un vector de complexes o *doubles*:

$$\{s,s,s,(s-2),(s-2),(s-3)\} \quad \rightarrow \quad \{s,(s-2),(s-3)\}$$

I finalment, en últim lloc, es troba el mètode *collectComplex*. Aquest mètodes ajunta dos vectors de complexes entrats com a paràmetres d'entrada en un de sol:

$$\begin{matrix}\{(s-2),(s-3)\}\\ \{(s-1),(s-3)\}\end{matrix}\Big|\rightarrow \quad \{(s-1),(s-2),(s-3),(s-3)\}$$

G.1.5. Adaptació per a representacions

Aquest mètodes serveixen per preparar certes dades per poder ser representades dins EJS, degut a que moltes representacions gràfiques demanen vectors de dades del tipus *double* i en el cas de TF disposa les dades en vector de complexes.

El primer conjunt, els mètodes *polsToDouble* i *zerosToDouble*, serveixen per obtenir els pols o zeros de la funció de transferència en format del tipus *double* o transformar objectes de la classe *Complex* a dades del tipus *double* de forma que puguin ser representats mitjançant els elements de representació gràfica del EJS.

El segon mètode que es pot trobar és el *simplePolsToDouble* i *simpleZerosToDouble* aquests dos mètodes retornen les arrels del sistema en matrius del tipus *double* però únicament donen els valors de les arrels una vegada, no retorna multiplicitat. En altres paraules, retorna els punts on hi ha arrels i aquests valors no poden estar repetits.

G.1.6. Tests de causalitat

Els tests de causalitat (*causalityTest)* serveixen per comprovar que la funció de transferència sigui causal, es a dir, que el grau del denominador sigui més gran que el grau del numerador.

Aquest mètodes, retornen un booleà, que indica si és causal o no, essent *false* en cas de que no sigui una funció de transferència causal i serveixen per tal de saber internament si es poden realitzar certes operacions o no i en cas de que no es pugui, indicar a l'usuari per a quin motiu es degut.

G.1.7. Mètode per a obtenció de continuïtat en els resultats de fases

El següent conjunt, els mètodes *degCont* i *radCont,* s'utilitzen per mantenir continuïtat gràfica per les dades que serveixen per representar fases. Explicat amb més detall, s'ha de comentar que certs mètodes, com ara la resposta freqüencial, representació del Bode, etc. Utilitzen funcions ja definides en Java que calculen angles (en radiants o graus) i que els seus resultats estan compresos entre -180º i 180º. Aquest fet, fa que en moltes representacions gràfiques hi aparegui una discontinuïtat com la que es pot observar tot seguit:

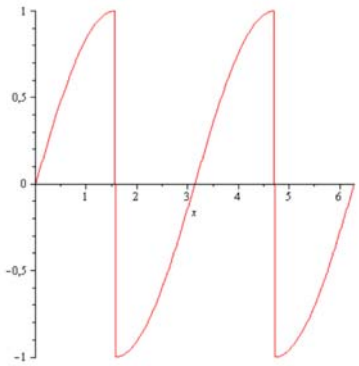


Fig. G.2. Discontinuïtat de la fase.

Aquest mètodes, si se'ls dóna com a paràmetre d'entrada el vector de dades que representa la fase a dibuixar, retorna un vector de la mateixa dimensió amb dades que representen la mateixa informació però de forma contínua:

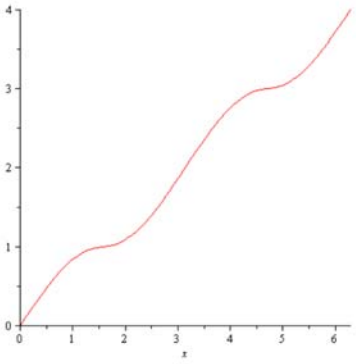


Fig. G.3. Continuïtat de la fase.

G.1.8. Comprovació d'igualtat d'objectes de la classe TF

Aquest mètode (*isEqual*) serveixen per comprovar si dos objectes de la classe TF són iguals o no, retornant un booleà amb el valor de *true* en cas afirmatiu.

H. Funcionament de les aplicacions EJS

Com es pot observar a la següent imatge, l'aplicació està dividida en dos pantalles. La pantalla de l'esquerre, serveix per modificar els elements de la funció de transferència (pols, zeros, guany, etc.). Mitjançant l'editor de pols i zeros i el caixetins de sobre es poden anar configurant les funcions de transferència, podent veure representat en temps real les funcions de transferència de sistema resultant a la part superior esquerre.

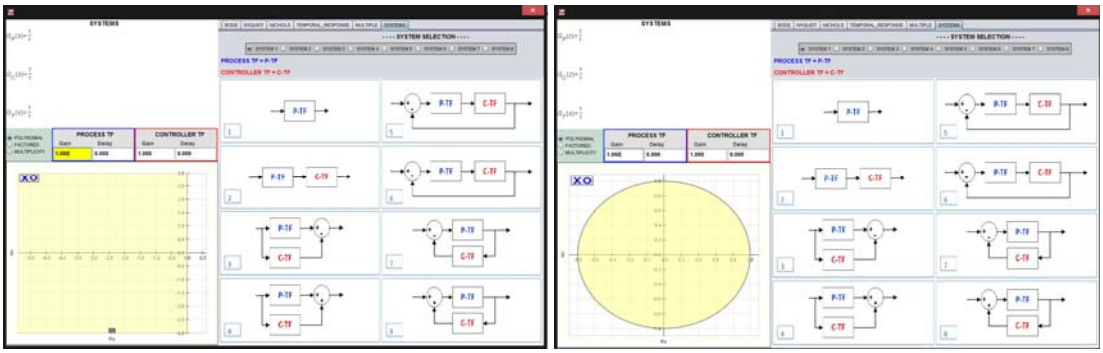


Fig. H.1. Vista general de les aplicacions EJS

La part dreta de la pantalla són els diferents elements de representació de resultats i de selecció d'opcions per a aquests resultats.

Per a seguir un ordre, tot seguit es mostrarà els passos que s'hauria d'anar seguint en els següents apartats.

H.1. Selecció de funció de transferència

Primer de tot s'hauria de seleccionar el sistema amb el qual es vol treballar.

Aquestes aplicacions incorporen 8 sistemes predefinits podent triar treballar únicament amb un al moment. Cada sistema està identificat amb un número al seu costat inferior esquerre (per tal de relacionar-lo amb el botó de la seva tria).

Per a triar un dels sistemes, cal anar a la pestanya *Systems*. Allà, hi ha representats els 8 sistemes a triar. Per a sel·leccionar-los, cal clicar-ne el desitjat a la part superior anometat com a "*System selection*" i clicar sobre el que es vol.

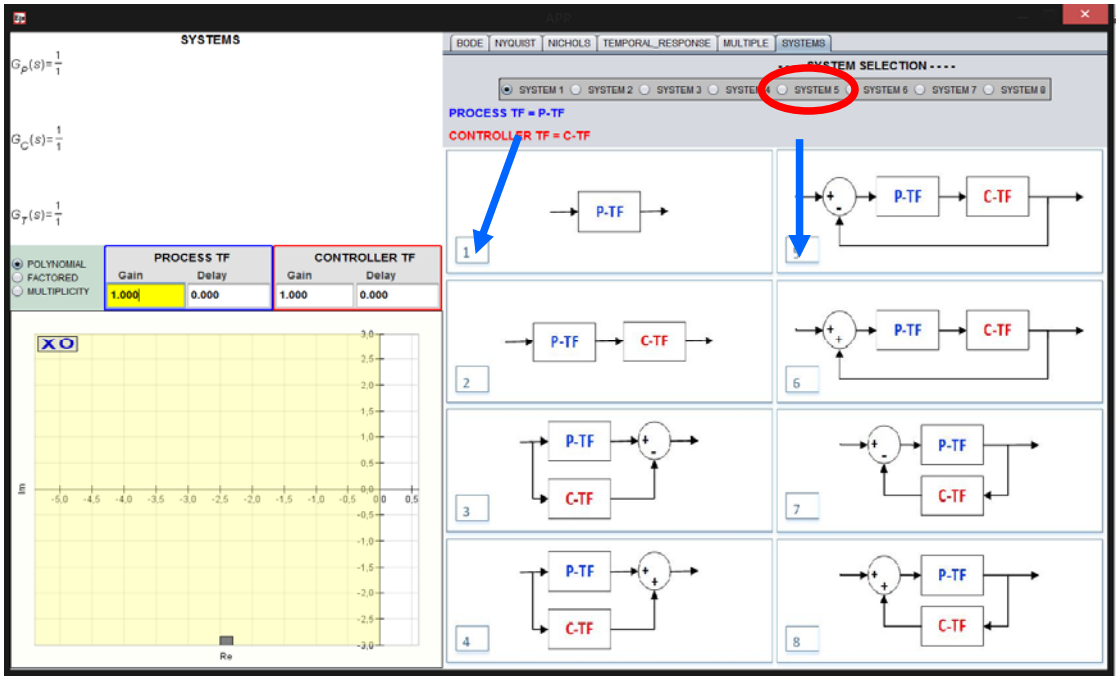


Fig. H.2. Aplicació EJS: selecció del sistema

Aquest tipus de selecció és un concret dels molts que es podria dissenyar, com ara, enlloc de *bullets* fer servir un desplegable amb el número del sistema o un desplegable sense les imatges o que fos un sistema fix per aplicaió i llavors fer una aplicació per sistema, etc. Tenint d'aquesta manera i gran quantitat de possibilitat de definició de l'aplicació.

H.2. Modificació els elements del sistema

Posant d'exemple haver triat el systema 5.

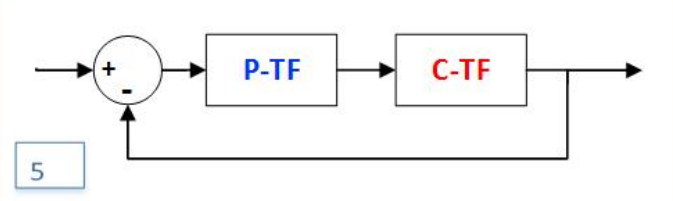


Fig. H.3. Aplicació EJS: sistema 5.

Tal i com s'indica a la mateixa aplicació, P-TF (color blau) es tracta del process del sistema i C-TF (color vermell) es tracte del controlador del sistema.

Per a distingir-los i relacionar-los a l'editor de pols i zeros i a les caselles de definició d'elements, s'ha fet servir un codi de colors. Per tant, sempre que es modifiqui elements de color blau, s'està treballant amb el procés del sistema. I, sempre que es modifiquin elements de color vermell, s'està modificant el controlador.

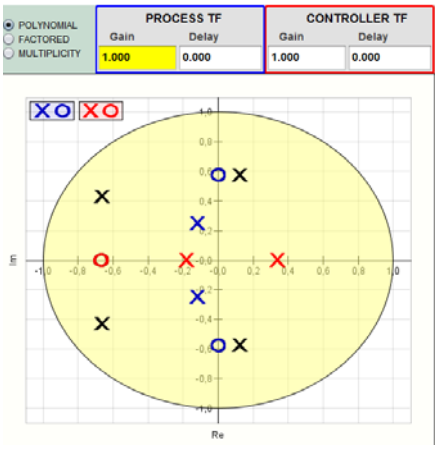


Fig. H.4. Aplicació EJS: elements i colors.

El pols i zeros que apareixen de color negre, es tracten dels pols i zeros del sistema complet en conjunt. Es a dir, no es poden modificar i van variant a mesura que modifiquem les parts individualment.

H.3. Escriptura de la funció de transferència

A mesura que es van modificant els elements del sistemes, les funcions de tranferencia (tant del procés, com controlador i sistema) es van escrivint a la part superior de la part esquerre de l'aplicació. A més a més, es pot seleccionar quin tipus d'escriptura es vol representar fent servir el panell de color verd de la pantalla de l'esquerre (podent triar a l'apliació entre forma polinòmica, factoritzada i factoritzada detectant la multiplicitat):

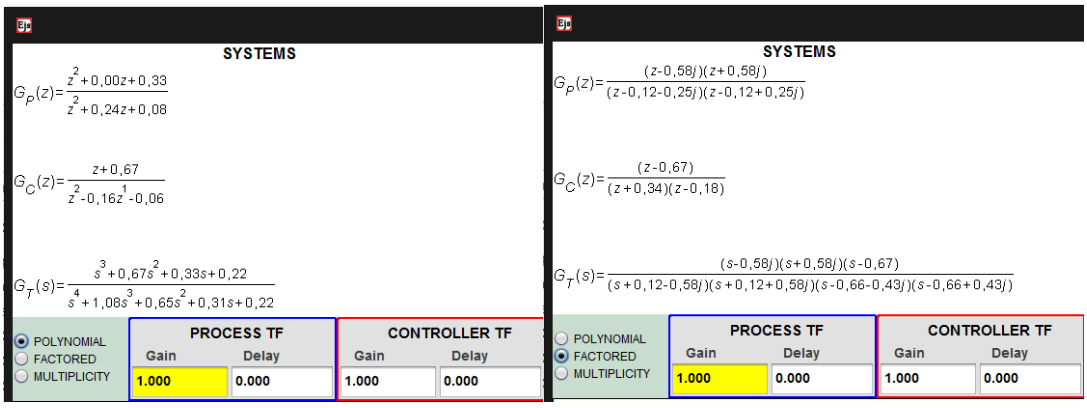


Fig. H.5. Aplicació EJS: escriptura dels sistemes (polinòmica i factoritzada).

Per identificar cada element, es fa a partir del subíndex de la escriptura, on P és el procés, C és el controlador i T el sistema total.

H.4. Representació de Nyquist i Nichols

Les aplicacions EJS permeten modificar els elements del sistema i ananr veient en temps real com les gràfiques representades van variant. Uns dels casos, és amb els diagrames de Nyquist i Nichols.

Per poder veure els diagrames de Nichols i Nyquist, s'ha de seleccionar una de les dos pestanyes de la part dreta superior. Quan s'hagi definit un sistema (o es vagi modificant els elements d'un) les zones de representació mostraran els diagrames i com aquests van variant.

També a la part inferior dels diagrames, hi ha una barra horitzontal modificable. Aquesta barra indica el nombre de punts a representar en els diagrames, on, menys punts (part esquerra de la barra) significa menys precisió però més velocitat de representació i més punts (part dreta de la barra) significa més temps de càlcul i més presició. En aquest punt, l'usuari pot triar el nombre de punts adequat segons vegi els resultats (a partir de cert nombre de punts la millora en resolució no es nota).

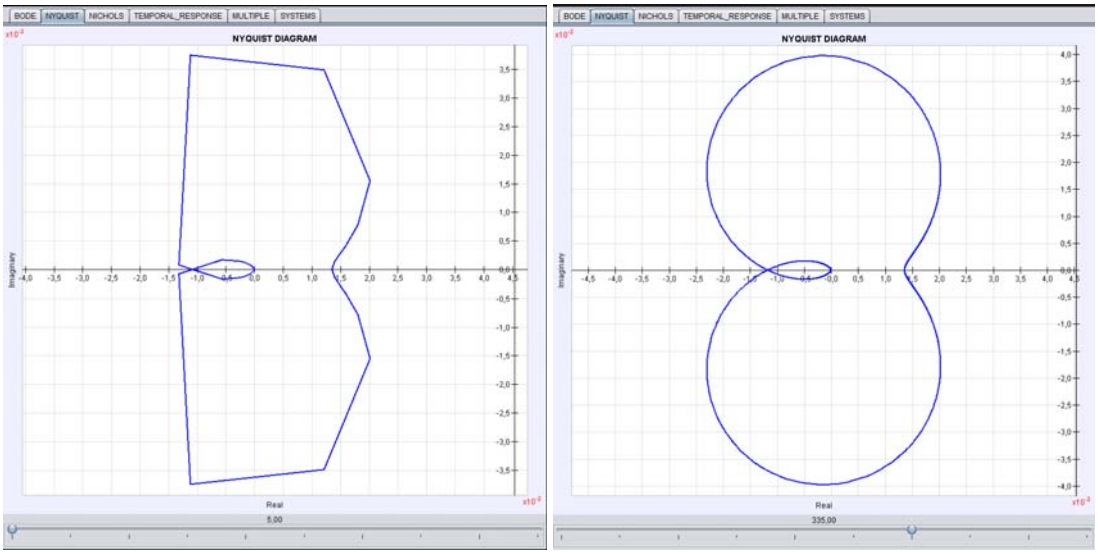


Fig. H.6. Aplicació EJS: exemple de definició de resolució en Nyquist.

H.5. Representació del diagrama de Bode

Per poder veure el diagrama de Bode, una vegada definit un sistema, es selecciona la pestanya amb el nom de "Bode" de la part dreta de l'aplicació, allà es pot veure la representació del diagrama de Bode.

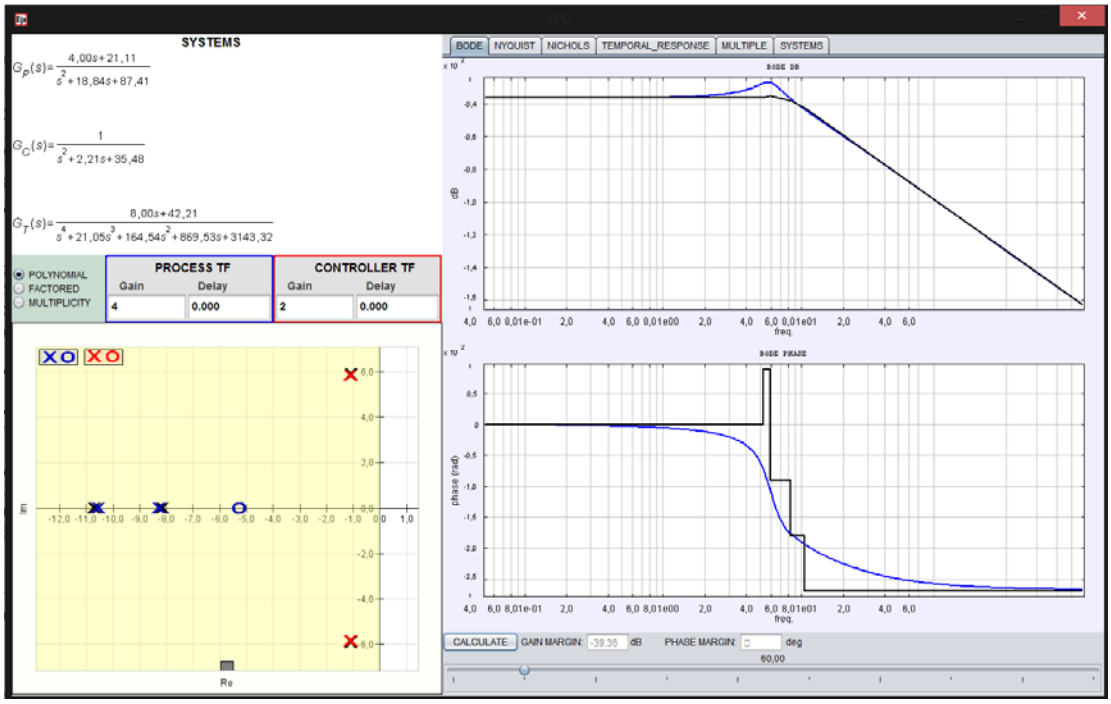


Fig. H.7. Aplicació EJS: exemple de diagrama de Bode

A l'aplicació EJS en temps continu (CTF), en aquesta pestanya hi haurà representat el diagrama de bode real i assimpotòtic. A l'aplicació en temps discret (DTF), únicament el bode.

A la part inferior de la pantalla hi ha un botó amb el text "CALCULATE" que clicant-lo, serveix per calcular el marge de guany i el marge de fase del sistema. S'ha fet d'aquesta forma, per dos motius: per mostrar diferents opcions de mostrar informació dins l'aplicació i per evitar excés de càlculs al modificar qualsevol element.

També, hi ha a la part inferior de tot de la pantalla del diagrama una barra horitzontal modificable. Aquesta barra indica el nombre de punts a representar en el diagrama de bode, on, menys punts (part esquerra de la barra) significa menys precisió però més velocitat i més punts (part dreta de la barra) significa més temps de càlcul i més presició. En aquest punt, l'usuari pot triar el nombre de punts adequat segons vegi els resultats (com en el cas anterior, a partir de cert nombre de punts la millora en resolució no es nota).

H.6. Representació múltiple

S'ha habilitat una pestanya per poder mostrar varis diagrames alhora. Aquesta pestanya, anomenada "Multiple" demostra que mitjantçant la llibreria i el EJS hi ha la possibilitat de fer aplicacions interactives, en temps real, personalitzables i que permet un gran ventall d'opcions.

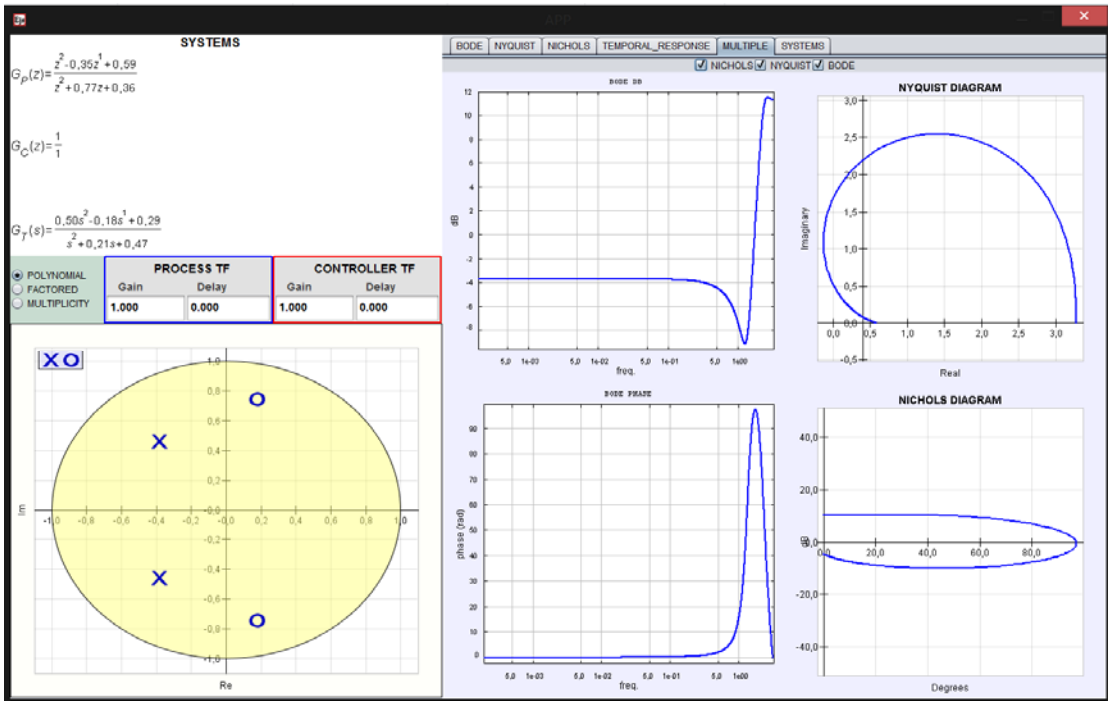


Fig. H.8. Aplicació EJS: exemple de múltiples gràfics

Com es pot veure, en aquesta pestanya s'hi troba els diagrames de Bode, Nyquist i Nichols tots junts. A més, existeix la opció de triar quins diagrames es mostren i quins no amb el caixetins superiors.

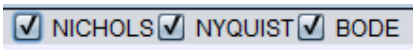


Fig. H.9. Aplicació EJS: selecció de gràfiques visibles

H.7. Representació de resposta temporal

Per últim, tant en l'aplicació CTF com a l'aplicació DTF, es pot utilitzar la funcionalitat de representar la resposta temporal de les funcions de transferència definides a partir d'una funció d'entrada donada.

Per a fer-ho, cal primer seleccionar la pestanya "Temporal Response". Allà cal definir la funció de transferència amb les que es vol treballar (mitjançat l'editor de pols i zeros). Fet això, s'ha de triar la entrada amb la qual es vol representar la resposta remporal i finalment, definir el nombre de mostres (resolució del resultat) i temps en el qual es vol obtenir aquesta representació fent servir les opcions de l'apartat superior.

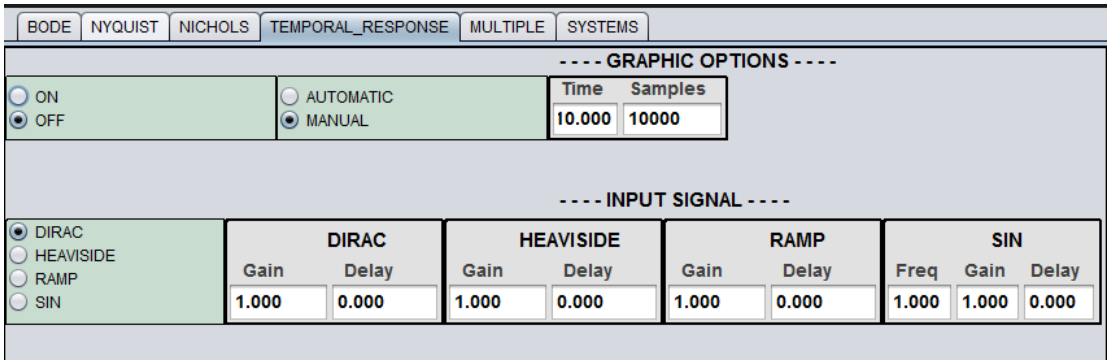


Fig. H.10. Aplicació EJS: opcions de la resposta temporal.

Després d'haver triat totes les opcions que es volen. S'ha de clicar a "ON" i en aquest moment l'aplicació passa a fer el càlucl de la resposta temporal. En estat de "ON" la representació passa a ser interactiva com totes les altres, però en aquest cas s'ha deixat l'opció de poder aturar els càlculs per poder evitar excés de processament i mostrar també més opcions de l'aplicació.

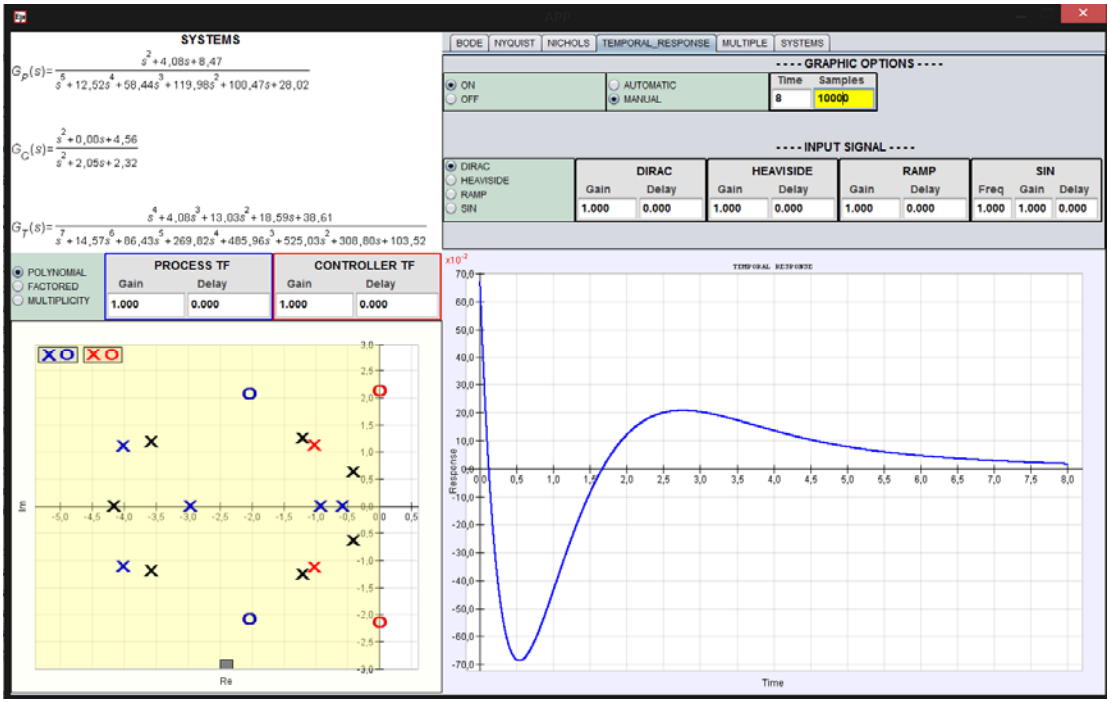


Fig. H.11. Aplicació EJS: resposta temporal davant una delta.

I. Codis d'integració de les aplicacions EJS

Tal i com s'ha comentat a la memòria principal del projecte, es poden distingir diferents codis dins l'apliació EJS.

Al ser les aplicacions eines que surten de l'abast del projecte però que s'han realitzat per demostrar el seu funcionament, ventall de possibilitats i una demostració més visual del què pot fer el projecte, tot seguit es mostraran els codis que vinculen únicament la llibreria Java creada amb l'aplicació EJS. La resta de codis, són propis de cada aplicació i no servirien com a representació dels resultat obtinguts del projecte, a diferencia de les codis d'integració, que sí demostren la possible interconnexió entre els dos elements (EJS i llibreria del projecte).

L'explicació serà molt breu i únicament intenta servir d'exemple de la integració de codi d'una aplicació EJS (és a dir, ve a ser unicament exemple).

I.1. Escriptura de fórmules

```
if(factoredForm){
  formula = tf.toFactoredForm("G","P");
  formulaC = tfC.toFactoredForm("G","C");
  formulaT = tfT.toFactoredForm("G","T");}
else if(multiplicityForm){
  formula = tf.toMultiplicityForm("G","P");
  formulaC = tfC.toMultiplicityForm("G","C");
  formulaT = tfT.toMultiplicityForm("G","T");}
else{
  formula = tf.toPolynomialForm("G","P");
  formulaC = tfC.toPolynomialForm("G","C");
  formulaT = tfT.toPolynomialForm("G","T");}
```

L'escriptura de fórmules s'ha adaptat de forma que amb un sol mètode cridat es retorni un *String* que es guarda en una variable i ja és la fórmula escrita en Latex que es mostrada amb un dels elements de representació del EJS.

I.2. Guany i retard de les funcions de transferència

```
tf.setGain(gainP);
tfC.setGain(gainC);
tf.setDelay(delayP);
tfC.setDelay(delayC);
```

Bàsicament aquesta part agafa les variables que es modifiquen en els panells de guany i retard de la funció del procés i el controlador i els guarda als objectes CTF i DTF.

I.3. Diagrama de Bode

```
double[][] bode = tfx.assymptoticBode();
assimptBodeX = new double[(int) bode[0].length];
assimptBode = new double[(int) bode[0].length];
assimptPhase = new double[(int) bode[0].length];
this.assimptBodeX = bode[0];
this.assimptBode = bode[1];
this.assimptPhase = bode[2];

tfx.setBodePoints(numPoints);
bode = tfx.bode();
  tfx.resetBodePoints();
bodeX = new double[(int) bode[0].length];
bodeDB = new double[(int) bode[0].length];
bodePhase = new double[(int) bode[0].length];
bodeX = bode[0];
bodeDB = bode[1];
bodePhase = bode[2];
if(pmCalculate){
  gainMargin = tfx.getGainMargin();
  phaseMargin = tfx.getPhaseMarginRad(); }
```

Es diagrames de representació s'alimenten de les dades a mostrar a l'eix Y i en quina posició de l'eix X s'han de mostrar. Els mètodes de Bode i Bode Assimpòtic retornen les dades dels eixos X i Y directament, de forma que s'assignin a variables que EJS fa servir pels gràfics. Aquesta part, també afegeix dins un condicional *if* que si s'ha seleccionat, faci el càlcul de marge de guany i marge de fase (quan s'apreta el boto, fa el càlcul).

I.4. Diagrama de Nyquist i Nichols

```
tfx.setBodePoints(numPoints);
double[][] nichols = tfx.nichols();
nicholsXP = new double[(int) nichols[0].length];
nicholsYP = new double[(int) nichols[0].length];
tfx.resetBodePoints();
for(int i =0; i<nichols[0].length; i++){
```

```
            nicholsXP[i] = nichols[0][i];  }
  for(int i =0; i<nichols[0].length; i++){
           this.nicholsYP[i] = nichols[1][i];  }
```

La part de Nyquis és exactament igual, però cambiant el mètode nichols pel mètode Nyquist, Aquest mètode agafa el valor que l'usuari ha indicat a la barra de resolució i a partir d'aquest, calcula els punts a representar dels diagrames, els quls, els guarda dins les variables que EJS fa servir per representar els gràfics.

I.5. Resposta temporal

```
if(!onoff){
 time = new double[1];
    time[0] = 0;
 trY = new double[1];
    trY[0] = 0;}
if(onoff){
//FUNCTION INTRO SELECTION
if(dirac){
  tfIntro = new els.CTF().diracDelta(gainIntro,delayIntro);}
else if(heaviside){
  tfIntro = new els.CTF().heaviside(gainIntro,delayIntro);}
else if(ramp){
  tfIntro = new els.CTF().ramp(gainIntro,delayIntro);}
else if(sin){
   tfIntro = new els.CTF().sin(freqIntro,gainIntro,delayIntro);}
//TIME CALCULATION
if(!chooseTime){
    time = new double[10000];
    time[0] = 0;
    for(int i=1; i<samplesNumber;i++){
       time[i] = time[i-1]+50/samplesNumber;    }
     this.trY = tfx.temporalResponse(tfIntro,time);  }
if(chooseTime){
  if(samplesNumber <= 0){
     samplesNumber = 1;   }
    time = new double[samplesNumber];
    time[0] = 0;
    for(int i=1; i<samplesNumber;i++){
       time[i] = time[i-1]+timeToShow/samplesNumber;    }
```

```
this.trY = tfx.temporalResponse(tfIntro,time);  } }
```

Tot i ser el codi més llarg, es tracta d'un codi pot complex. Els passos que realitza bàsicament són: si la opció de *OFF* està activada, mostra un punt no visible al gràfic. Si està activada, crea la funció d'entrada segons quina hagi triat l'usuari i els paràmetres indicats. Tot seguit crea el vector de punts a representar en el temps de la resposta temporal i finalment, els calcula per guardar-los dins la variable que util·litza EJS per crear el diagrama.

Bibliografia

Referències bibliogràfiques

[1] ORACLE. Technology Network. Resources for Java Developers. [http://www.java.com/es/, 28/01/2013]

[2] UCL. DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING. Michael Thomas Flanagan's Java Scientific Library [http://www.ee.ucl.ac.uk/~mflanaga/java/, 08/06/2012]

[3] THE ECLIPSE FOUNDATION INC. Ottawa, Ontario, Canada. [http://www.eclipse.org/, 28/01/2013]

[4] UNIVERSIDAD DE MURCIA. Departamento de matemáticas. Ejs Wiki. 2011. [http://www.um.es/fem/EjsWiki/Es/HomePage, 10/06/2012]

[5] Ogata, Katsuhiko. Ingeniería de control moderna. 3ª ed. México D.F: Prentice Hall, 1998. ISBN 9701700481

[6] Ogata, Katsuhiko. Sistemas de control en tiempo discreto. 2ª ed. México: Prentice Hall, 1996. ISBN 9688805394.

[7] LaTeX Project. A document preparation system. [http://latex-project.org/, 04/06/2013]

Bibliografia complementària

[8] FACTORES DE EMISIÓN DE CO2 Y COEFICIENTES DE PASO A ENERGIA PRIMARIA DE DIFERENTES FUENTES DE ENERGIA FINAL CONSUMIDAS EN EL SECTOR DE LOS EDIFICIOS EN ESPAÑA. Ministerio de Industria, Comercio y Turismo.
http://www.minetur.gob.es/energia/desarrollo/EficienciaEnergetica/RITE/propuestas/Documents/2014_03_03_Factores_de_emision_CO2_y_Factores_de_paso_Efinal_Eprimaria_V.pdf